Mathématiques en liaison avec des problèmes concrets (Niveau terminale scientifique et première année d'université)

Fernand Didier, Jacques Gispert, Dominique Proudhon,
Robert Rolland et Patrick Soubeyrand

Volume I

F. Didier, J. Gispert, D. Proudhon, R. Rolland, P. Soubeyrand
IREM d'Aix-Marseille, Luminy Case 901, F13288 Marseille CEDEX 9
e-mail : didier@irem.univ-mrs.fr, Jacques.Gispert@lidil.univ-mrs.fr
dproudhon@wanadoo.fr, rolland@iml.univ-mrs.fr
soubeyrand.patrick@free.fr

Table des matières

1 Introduction

Les documents, exercices et problèmes qui suivent ont été élaborés dans le cadre de la liaison **Lycées-Universités**. Ce sont donc, pour les élèves des terminales scientifiques, des questions difficiles qui peuvent servir de thèmes de réflexion, ou même de directions de travail pour des projets. Ils peuvent être donnés aussi dans l'enseignement supérieur, comme applications directes du cours. Au-delà de l'utilisation directe de ces développements dans telle ou telle classe, il nous semble que beaucoup de ces applications et les approches qui en sont faites font partie de la culture scientifique utile, sinon indispensable, aux professeurs de mathématiques, culture sans laquelle les cours deviennent "clairs comme un bouillon dans lequel a trempé un os sans moëlle". Ainsi une partie des thèmes proposés ne sont pas à proprement parler dans quelque programme scolaire que se soit, ni même d'ailleurs parfois dans les cours universitaires. On y présente des questions scientifiques et on y parle peu de techniques de la classe, toutes choses qui pourraient laisser penser que ce travail est inintéressant, lors d'une première analyse superficielle, alors que le public visé est celui des professeurs de mathématiques des collèges et lycées. Mais si on veut bien considérer que le rôle du professeur est de dispenser une culture, au-delà des techniques de sa discipline, alors ces textes pourront, nous l'espérons, se révéler d'un certaine utilité.

Nous avons choisi de développer des sujets liés à des questions concrètes. Ce faisant notre but est de montrer des interventions des outils mathématiques dans la compréhension et la résolution de problèmes posés par les sciences expérimentales ou plus généralement par les besoins des techniques.

Suivant les divers thèmes que nous avons choisis, la forme diffère : certaines parties sont sous forme d'exposés, d'autres de problèmes ou d'exercices.

Remerciements : Ce document a été réalisé en LaTeX, système de macro-instructions au dessus de TeX. Nous remercions Donald Knuth (auteur de TeX) et Leslie Lamport (auteur de LaTeX) ainsi que les nombreux développeurs de paquetages qui ont fourni cet extraordinaire outil d'édition indispensable aux mathématiciens.

Les essais informatiques ont été faits avec le logiciel xcas, en simulation Maple. Nous remercions Bernard Parisse auteur du logiciel.

Première partie

Principe de moindre action

2 Présentation

Nous avons choisi comme premier thème, pour cette suite de documents, le principe de moindre action, qui s'applique en optique, en mécanique et qui a sur le plan de l'histoire des sciences une importance capitale. Le principe de **moindre action** a été énoncé par Pierre-Louis Moreau de Maupertuis (1698-1759) en 1744 dans un article, paru dans les Mémoires de l'Académie des Sciences, intitulé "Accord des différentes lois de la nature qui avaient jusqu'ici paru incompatibles". Ce travail généralisait le principe de "trajet de temps minimal", formulé en 1657 par Pierre de Fermat (1601-1665), au sujet de la loi de la réfraction. La loi de la réfraction avait été énoncée auparavant par Willebrord Snel van Royen (1591-1626), et par René Descartes (1596-1650). Cependant, avant le travail de Fermat, cette loi n'apparaissait pas comme provenant d'un minimum de temps de parcours. Le mérite du travail de Maupertuis, dans la continuité de celui de Fermat, fut de montrer, à travers cette idée de minimum, un lien entre les lois de l'optique et celles de la mécanique. On rejoint ainsi le calcul des variations de Isaac Newton (1642-1727), Leonhard Euler (1703-1783) et plus tard Joseph-Louis Lagrange (1736-1813).

La présentation est faite sous forme d'exercices avec corrigés.



Voici un timbre remémorant la mesure de l'arc de méridien en Laponie par Maupertuis.

Les problèmes qui suivent partent des lois de la réflexion et de la réfraction, en montrant bien comment elles s'interprêtent en terme de minimum de temps de parcours. Puis on regarde ce qu'il se passe quand l'indice de réfraction

varie continument. Les problèmes d'optique rejoignent alors des problèmes de recherche de trajectoires en mécanique.

3 Distance d'un point aux points d'une droite

Cet exercice est un simple petit calcul de distance qui sera utile dans les exercices suivants.

1. Dans un repère orthonormé (O, x, y) on considère un point A de coordonnées (x_0, y_0). On étudie la fonction d_A qui à tout x associe la distance de A au point M de l'axe (Ox) d'abscisse x.

a) En utilisant le théorème de Pythagore on étudiera les variations de la fonction d_A et en particulier les extrema et le comportement en $+\infty$ et $-\infty$.

b) Soit $\delta \in \mathbb{R}^+$. Chercher les valeurs de x pour lesquelles $d_A(x) = \delta$.

2. Dans un repère orthonormé (O, x, y) on considère un point A de coordonnées (x_0, y_0) avec $y_0 \neq 0$. On étudie la fonction d_A qui à tout x associe la distance de A au point M de l'axe (Ox) d'abscisse x.

a) Calculer la dérivée de d_A et étudier les variations de cette dérivée.

b) Étudier les variations de la fonction d_A.

c) Calculer

$$\lim_{x \to \infty} \left(d_A(x) - (x - x_0) \right).$$

En conclure l'existence d'une asymptote en $+\infty$ dont on calculera l'équation. Quelle est la position de la courbe représentative de d_A par rapport à cette asymptote.

c) Que se passe-t-il en $-\infty$?

4 Réflexion

Nous établissons ici la loi de la réflexion en calculant un minimum de chemin parcouru.

3. Dans un repère orthonormé (O, x, y) on considère deux points A et B de coordonnées respectives $(0, y_A)$ et (x_B, y_B) avec $y_A, x_B, y_B > 0$. Le point H sera le point de l'axe (Ox) d'abscisse x_B.

On étudie la fonction d_{AB} qui à tout x associe la somme des distances de A et de B au point M de l'axe (Ox) d'abscisse x.

a) Calculer la dérivée de d_{AB} et montrer que cette dérivée s'annule en un point x_0 et un seul. De plus on montrera que ce point appartient à l'intervalle $]0, x_B[$.

b) Étudier les variations de d_{AB}.

c) On note I le point de l'axe (Ox) d'abscisse x_0. Montrer que

$$(\overrightarrow{IO}, \overrightarrow{IA}) = -(\overrightarrow{IH}, \overrightarrow{IB}).$$

5 Réfraction

C'est au tour maintenant de la loi de la réfraction concernant un rayon lumineux qui passe d'un demi-espace ayant un certain indice à l'autre demi-espace ayant un autre indice. Rappelons que l'indice d'un milieu est $\frac{c}{v}$ où c est la vitesse de la lumière dans le vide et v la vitesse de la lumière dans le milieu considéré.

4. Dans un repère orthonormé (O, x, y) on considère deux points A et B de coordonnées respectives $(0, y_A)$ et (x_B, y_B) avec $y_A, x_B > 0$ et $y_B < 0$. On introduit le point M de l'axe (Ox) d'abscisse x. Un mobile ayant une vitesse v_1 quand il se déplace dans le demi-plan $y \geq 0$, et une vitesse v_2 dans le demi-plan $y < 0$, va de A en B en passant par M (il décrit les segments $[AM]$ et $[MB]$).

a) Calculer le temps $T_{AB}(x)$ mis par le mobile pour faire ce parcours.

b) Calculer la dérivée de T_{AB} et montrer que cette dérivée s'annule en un point x_0 et un seul. De plus on montrera que ce point appartient à l'intervalle $]0, x_B[$.

c)Étudier les variations de T_{AB}.

d) On note I le point de l'axe (Ox) d'abscisse x_0. Montrer que

$$\frac{1}{v_1} sin(\overrightarrow{OA}, \overrightarrow{IA}) = \frac{1}{v_2} sin(\overrightarrow{AO}, \overrightarrow{IB}).$$

6 Indice variable

Dans les exercices précédents on a minimisé le temps de trajet pour aller d'un point à un autre. En optique par exemple où l'indice d'un milieu est

défini comme étant le rapport $\frac{c}{v}$ où c est la vitesse de la lumière dans le vide et v la vitesse de la lumière dans le milieu considéré, la loi de la réfraction au passage entre deux milieux d'indices respectifs n_1 et n_2 s'exprime par

$$n_1 sin(i_1) = n_2 sin(i_2)$$

ainsi qu'on l'a vu dans la section 5.

On peut alors regarder ce qu'il se passe si on fait varier continûment l'indice en fonction de l'abscisse du point ou en fonction de l'ordonnée du point. Dans ces cas on peut supposer d'après l'étude faite précédemment que la trajectoire devra vérifier $nsin(i) = Constante$ avec $0 < i < \frac{\pi}{2}$ et $Constante > 0$. Voici divers exercices basés sur ce principe.

6.1 Quelques exemples

5. Soit f une fonction définie sur $[0, +\infty[$ dérivable telle que $f(0) = 0$ et $f'(0) = \tan(i_0) > 0$. On appelle i l'angle entre l'axe Ox et la tangente au point d'abscisse x au graphe de f, de telle sorte que $\tan(i)$ soit le coefficient directeur de cette tangente.

On suppose que $0 < i < \frac{\pi}{2}$ et que n est une fonction croissante et positive de x. Montrer que i est une fonction décroissante de x et en particulier que $i \leq i_0$.

5.A Des exemples avec i petit

On va supposer que l'angle i_0 est suffisamment petit pour que lorsque $0 \leq i \leq i_0$ on puisse avoir $\tan(i) \approx sin(i)$. Si bien que la condition $nsin(i) = C$ pourra être remplacée par $n\tan(i) = C$ (où C est une constante).

a) Cas où $n = n_0(1 + kx)$, $k > 0$. Calculer $f'(x)$ puis $f(x)$.

b) Cas où $n = n_0(2 - e^{-\alpha x})$, $\alpha > 0$. Calculer $f'(x)$ puis $f(x)$.

c) Refaire les mêmes calculs avec $n = \frac{n_0(\alpha x+1)}{x+1}$, $\alpha > 1$.

5.B Un exemple avec i pas si petit que ça

On se propose ici de faire une comparaison entre les résultat obtenus par la méthode exposée précédente et le calcul exact, dans un cas où celui-ci aboutit.
A cet effet, on notera f_e la solution exacte, et f_a la solution approchée.
De plus, on suppose que $n(x) = n_0\sqrt{ax + 1}$, avec n_0 et a strictement positifs.
Le lecteur pourra facilement contrôler que n est une fonction positive, croissante, lorsque x décrit $\mathbb{R}_+$.

a) Calcul exact.

1. Montrer que $f_e'(x) = \frac{C}{\sqrt{n^2(x)-C^2}}$.

2. En déduire :

$$f_e(x) = \frac{2C}{a\,n_0}\left(\sqrt{ax+1-\frac{C^2}{n_0^2}} - \sqrt{1-\frac{C^2}{n_0^2}}\right).$$

b) Calcul approché (même principe qu'au **5.A**).

1. Montrer que $f_a'(x) \approx \frac{C}{n_0\sqrt{ax+1}}$.

2. En déduire :

$$f_a(x) = \frac{2C}{a\,n_0}\left(\sqrt{ax+1}-1\right).$$

c) On pose $\Delta(x) = f_e(x) - f_a(x)$.

1. Montrer que la fonction Δ est croissante sur $\mathbb{R}_+$.

2. Montrer que Δ admet une limite réelle, que l'on précisera, en $+\infty$.

d) On pose, pour $x > 0$, $q(x) = \frac{f_e(x)}{f_a(x)}$. Montrer que q admet une limite réelle, que l'on précisera, en $+\infty$.

5.C Extensions

On se propose ici de généraliser un certain nombre de résultats dans le cas général. Il s'agit toujours, ici, de faire une étude comparative entre le résultat approché obtenu par la méthode exposée au **5.1** et le calcul exact, dans le cas général. Les hypothèses sur f,i et n restent les mêmes qu'au chapeau du paragraphe 5

Pour effectuer cette comparaison,on notera f_e la solution exacte, et f_a la solution approchée.

a) Calcul exact.

1. Montrer que $f_e'(x) = \frac{C}{\sqrt{n^2(x)-C^2}}$.

2. En déduire que

$$f_e(x) = C.\int_0^x \frac{du}{\sqrt{(n(u))^2 - (C)^2}}.$$

b) Calcul approché (même principe qu'au **5.1**).

1. Montrer que $f_a'(x) \approx \frac{C}{n(x)}$.

2. En déduire :

$$f_a(x) = C\int_0^x \frac{du}{n(u)}.$$

c) On pose $\Delta(x) = f_e(x) - f_a(x)$.

1. Calculer la dérivée de Δ.Montrer que la fonction Δ est strictement croissante sur $\mathbb{R}_+$.

2. En déduire que Δ admet une limite λ, en $+\infty$. Montrer que λ est un réel strictement positif ou $+\infty$.

d) On pose, pour $x > 0$, $q(x) = \frac{f_e(x)}{f_a(x)}$.

1. Calculer la dérivée de q.Montrer que la fonction q est décroissante sur $\mathbb{R}_+$, strictement décroissante si de plus n est strictement croissante sur $\mathbb{R}_+$.

2. En déduire que q admet une limite réelle μ comprise entre 1 et $\frac{1}{\sqrt{1-\left(\frac{C}{n_0}\right)^2}}$.

e) Etude au voisinage de zéro :
Montrer que Δ tend vers zéro, lorsque x tend vers zéro.

Montrer que q tend vers $\frac{1}{\sqrt{1-\left(\frac{C}{n_0}\right)^2}}$, lorsque x tend vers zéro.

f) On suppose dans cette question que $n(x) = n_0.e^{\alpha x}$, où α est un réel strictement positif.

1. Effectuer les calculs dans ce cas.

2. Montrer que, pour $C = 1 = \alpha$ et $n_0 = \sqrt{2}$, la limite μ de q en $+\infty$ est égale à :

$$\frac{\pi\sqrt{2}}{4}$$

Montrer que, dans les mêmes conditions,

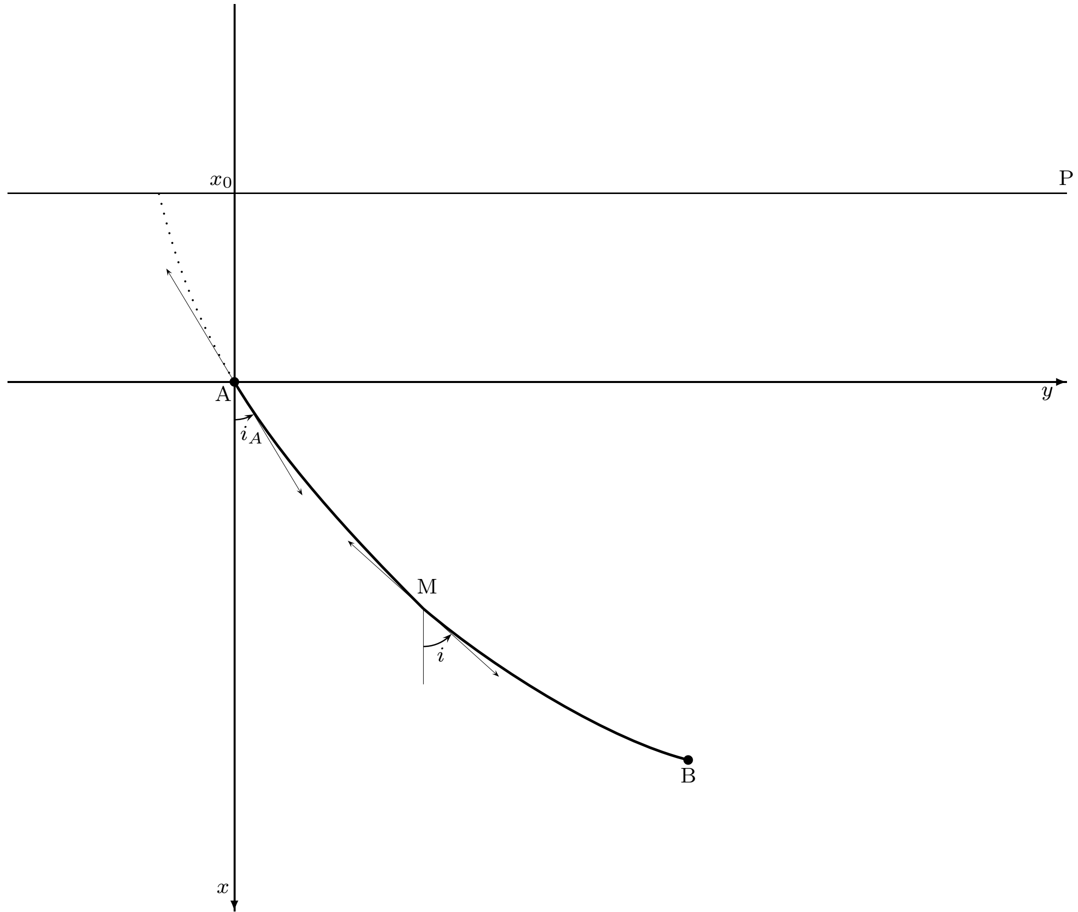
$$\lambda = \frac{\pi}{4} - \frac{1}{\sqrt{2}}$$

g) Montrer que si n tend vers $+\infty$, en $+\infty$, alors μ n'est pas forcément égal à 1.
On pourra s'intéresser au cas étudié à la question **f.**
Les fonctions f_e et f_a sont-elles équivalentes au voisinage de $+\infty$?

6.2 Profil d'une jetée, ou le problème de la dalle en pente

Le problème de la courbe de descente la plus rapide ou de la montée la plus difficile (d'où son utilisation pour le profil d'une jetée) a été posé par Galilée. La solution correcte de l'arc de cycloïde passant par les deux points qu'on veut joindre a été résolu par Newton, Leibnitz, L'Hospital, John et James Bernouilli. La solution de John and James Bernouilli utilise l'idée que nous reprenons, d'un temps de parcours de rayon lumineux. La courbe de descente la plus rapide est appelée courbe brachistochrone, et c'est donc un arc de cycloïde. Il se trouve que cette courbe est aussi tautochrone (montré par Huygens), c'est-à-dire que quel que soit la position du point de départ, le mobile arrive au point le plus bas de l'arche de cycloïde dans le même temps (ce que nous n'étudierons pas ici).

6. On fixe 2 points A, B et un plan horizontal P (cf. figure). Une masse ponctuelle M soumise à la gravitation à l'exception de toute autre force, tombe du plan horizontal P. Elle passe en A et en B en suivant un profil qu'on cherche à définir de telle sorte que le temps mis par M pour aller de A à B soit minimum. On choisit un repère orthogonal d'origine A orienté comme indiqué sur la figure. L'angle $(\overrightarrow{Ax}, \overrightarrow{V})$ (où $\overrightarrow{V}$ est le vecteur vitesse) sera noté i. Au point A cet angle a pour valeur i_A. On admettra que la trajectoire cherchée peut s'écrire sous la forme cartésienne $y = f(x)$, avec f dérivable.



Rappelons que la vitesse de M d'abscisse x est

$$v = \sqrt{2g(x - x_0)}.$$

Compte tenu de l'introduction du paragraphe 6 on est amené à imposer la condition

$$n \sin(i) = Constante$$

où n est proportionnel à l'inverse de la vitesse, c'est-à-dire

$$n \sin(i) = n_A \sin(i_A),$$

$$\sin(i) = \sqrt{x - x_0}\, \frac{\sin(i_A)}{\sqrt{|x_0|}}.$$

a) Calculer $x = \phi(i)$.

b) Etablir que $y = f(\phi(i))$.

c) Calculer $f'(x)$ en fonction de i. En déduire la dérivée de la fonction $f \circ \phi$.

d) En déduire que

$$y = \frac{|x_0|}{\sin^2(i_A)} \int_{i_A}^{i} 2\sin^2(u)\,du.$$

e) Calculer la fonction $f \circ \phi$.

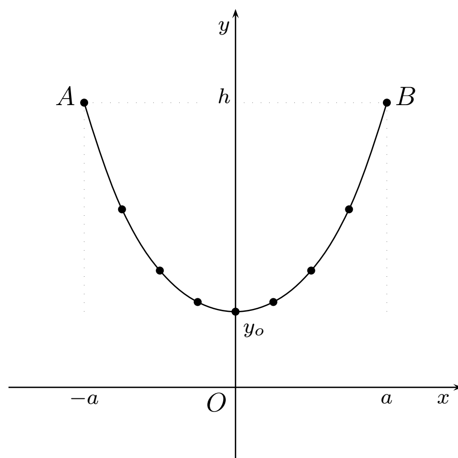
On a donc obtenu les deux coordonnées x et y du point M en fonction du même paramètre i.

f) Montrer que les coordonnées x et y du point M vérifient les relations

$$\begin{cases} x - x_0 = K(1 - \cos(2i)), \\ y - y_0 = K(2i - \sin(2i)). \end{cases}$$

g) Construire l'arc (AB) de cette courbe en prenant $x_0 = -1$, $i_A = \dfrac{\pi}{6}$ et $x_B = 2$.

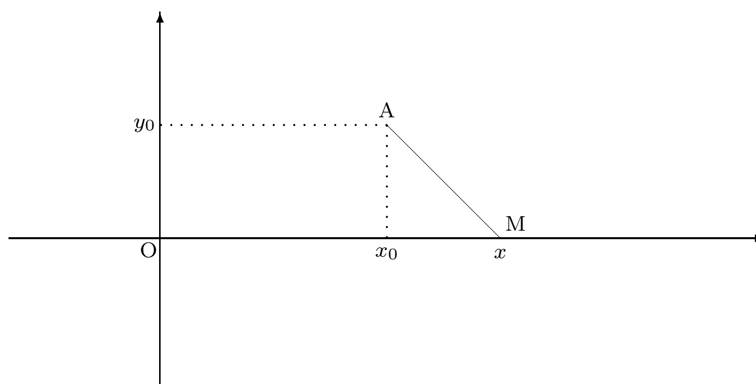
6.3 chaînette



7. Montrer que, lorsqu'on suspend un fil uniformément pesant, de longueur l, de masse $m = \rho l$, entre deux points A et B situés à la même hauteur h, il se dispose en forme de chaînette.
On supposera que l et h sont tels que le fil ne touche pas le sol.

7 Solution des exercices

Exercice 1.



La fonction d_A est définie sur $\mathbb{R}$ par : $d_A(x) = \sqrt{(x - x_0)^2 + y_0^2}$.

a) On remarque tout de suite que :

1. la fonction d_A est **minorée** : pour tout x réel, $d_A(x) \geq |y_0|$.

2. la courbe de la fonction d_A présente une **symétrie** par rapport à la droite d'équation $x = x_0$.

En effet, pour tout h réel, $d_A(x_0 + h) = d_A(x_0 - h)$.

3. la fonction d_A est **strictement croissante** sur l'intervalle $[x_0\ , +\infty[$.

4. enfin, pour tout x réel, $d_A(x) \geq |x - x_0|$. Il en découle : $\lim\limits_{x \to +\infty} d_A(x) = +\infty$.

On obtient donc le tableau de variation suivant :

x	$-\infty$	x_0	$+\infty$
$d_A(x)$	$+\infty$ ↘	$\lvert y_0 \rvert$	↗ $+\infty$

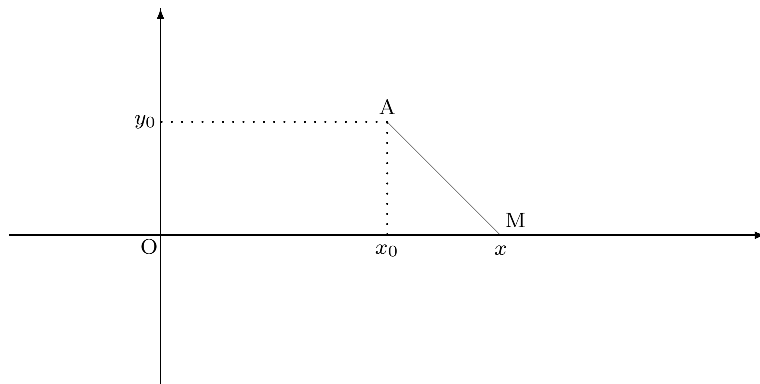
b) On déduit du tableau précédent que, δ étant un réel positif donné, l'équation $d_A(x) = \delta$:

- n'a pas de solution si $\delta < |y_0|$,

- a une solution ($x = x_0$) si $\delta = |y_0|$,

- a deux solutions distinctes x_1 et x_2 si $\delta > |y_0|$, x_1 et x_2 étant telles que $x_1 - x_0 = x_0 - x_2$.

Exercice 2.



La fonction d_A est toujours définie sur $\mathbb{R}$ par : $d_A(x) = \sqrt{(x-x_0)^2 + y_0^2}$, mais on rajoute la condition $y_0 \neq 0$.

a) y_0 n'étant pas nul, le radicande n'est jamais nul, et donc la fonction d_A est **dérivable sur** $\mathbb{R}$; et on obtient, pour tout x réel :

$$d'_A(x) = \frac{2(x-x_0)}{2\sqrt{(x-x_0)^2 + y_0^2}} = \frac{x-x_0}{\sqrt{(x-x_0)^2 + y_0^2}}.$$

La fonction d'_A est, elle-aussi, dérivable sur $\mathbb{R}$, et on a, pour tout x réel :

$$d''_A(x) = \frac{\sqrt{(x-x_0)^2 + y_0^2} - \dfrac{(x-x_0)^2}{\sqrt{(x-x_0)^2 + y_0^2}}}{(x-x_0)^2 + y_0^2} = \frac{y_0^2}{((x-x_0)^2 + y_0^2)^{\frac{3}{2}}} > 0.$$

La fonction d'_A est donc **strictement croissante** sur $\mathbb{R}$.

D'autre part, pour $x \neq x_0$,

$$d'_A(x) = \frac{x-x_0}{|x-x_0|\sqrt{1 + \dfrac{{y_0}^2}{(x-x_0)^2}}}.$$

D'où, si $x > x_0$, $d'_A(x) = \dfrac{1}{\sqrt{1+\dfrac{{y_0}^2}{(x-x_0)^2}}}$, et donc, $\lim\limits_{x\to+\infty} d'_A(x) = 1$.

Par contre, si $x < x_0$, $d'_A(x) = -\dfrac{1}{\sqrt{1+\dfrac{{y_0}^2}{(x-x_0)^2}}}$, et on a, $\lim\limits_{x\to-\infty} d'_A(x) = -1$.

On a donc le tableau de variation suivant :

x	$-\infty$	x_0	$+\infty$
$d'_A(x)$	-1 ↗	0 ↗	1

b) Des variations de d'_A on déduit son signe et donc le tableau de variation de d_A :

x	$-\infty$		x_0		$+\infty$
$d'_A(x)$		$-$	0	$+$	
$d_A(x)$	$+\infty$	↘	$\lvert y_0\rvert$	↗	$+\infty$

c) Pour $x \geq x_0$:

$$d_A(x)-(x-x_0) = \sqrt{(x-x_0)^2+y_0^2}-(x-x_0) = \frac{((x-x_0)^2+y_0^2)-(x-x_0)^2}{\sqrt{(x-x_0)^2+y_0^2}+(x-x_0)}$$

donc $$d_A(x)-(x-x_0) = \frac{y_0^2}{\sqrt{(x-x_0)^2+y_0^2}+(x-x_0)}.$$

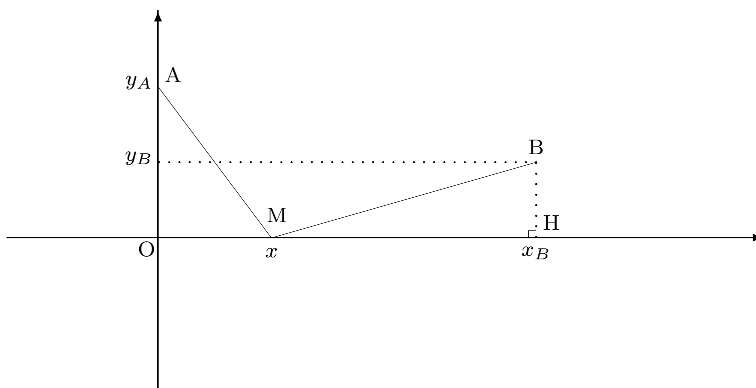
On constate que :

- d'une part $\lim_{x\to+\infty}[d_A(x)-(x-x_0)]=0$, ce qui prouve que la droite $\mathcal{D}$ d'équation $y=x-x_0$ est **asymptote oblique** en $+\infty$ à la courbe $\mathcal{C}$ de d_A.

- d'autre part, pour $x\geq x_0$, la différence $d_A(x)-(x-x_0)$ est >0, ce qui montre que, sur l'intervalle $[x_0\,,+\infty[$, la courbe $\mathcal{C}$ est **au-dessus** de la droite $\mathcal{D}$.

La courbe $\mathcal{C}$ étant **symétrique** par rapport à la droite d'équation $x=x_0$, on en déduit que la droite $\mathcal{D}'$, d'équation $y=-x+x_0$ est **asymptote oblique** en $-\infty$ à la courbe $\mathcal{C}$ de d_A et que, sur l'intervalle $]-\infty\,,x_0]$, la courbe $\mathcal{C}$ est **au-dessus** de la droite $\mathcal{D}'$.

Exercice 3.



La fonction d_{AB} est définie sur $\mathbb{R}$ par : $d_{AB}(x)=\sqrt{x^2+{y_A}^2}+\sqrt{(x-x_B)^2+{y_B}^2}$.

a) y_A et y_B n'étant pas nuls, les deux radicandes ne sont jamais nuls, et donc la fonction d_{AB} est **dérivable sur** $\mathbb{R}$; et on obtient, pour tout x réel :

$$d'_{AB}(x)=\frac{x}{\sqrt{x^2+{y_A}^2}}+\frac{x-x_B}{\sqrt{(x-x_B)^2+{y_B}^2}}.$$

On remarque que la fonction d'_{AB} est la somme de deux fonctions **strictement croissantes** sur $\mathbb{R}$. En effet, si on calcule les dérivées de ces fonctions, on va constater, comme on l'a déjà fait dans l'exercice **2.** (question **a**), que ce sont des réels >0.

On en déduit que la fonction d'_{AB} est **strictement croissante** sur $\mathbb{R}$.

D'autre part, x_B étant > 0, on a $d'_{AB}(0) < 0$ et $d'_{AB}(x_B) > 0$.

La fonction d'_{AB} étant **continue**, il découle du corollaire du théorème des valeurs intermédiaires que, dans l'intervalle $[0\,,x_B]$, l'équation $d'_{AB}(x) = 0$ possède une solution unique x_0.

De plus, du fait de la **monotonie** de d'_{AB}, si $x < 0$ alors $d'_{AB}(x) \leq d'_{AB}(0)$, donc $d'_{AB}(x) < 0$, et si $x > x_B$ alors $d'_{AB}(x) \geq d'_{AB}(x_B)$, donc $d'_{AB}(x) > 0$; ce qui montre qu'en dehors de l'intervalle $[0\,,x_B]$, l'équation $d'_{AB}(x) = 0$ ne possède aucune solution.

En résumé, la fonction d'_{AB} s'annule en un point x_0 et un seul, compris entre 0 et x_B.

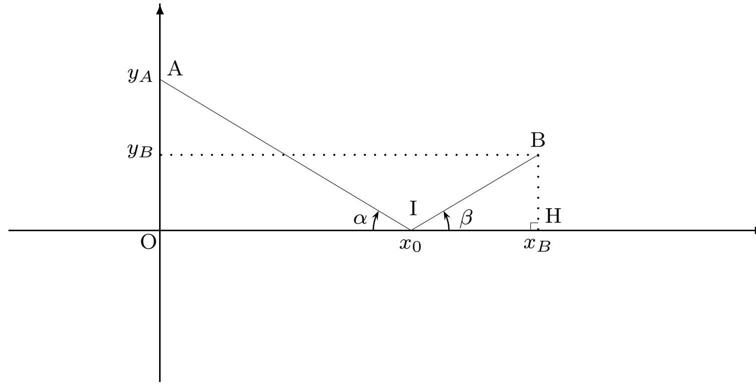
On en déduit le tableau de variation de la fonction d'_{AB} :

x	$-\infty$	x_0	$+\infty$
$d'_{AB}(x)$		0	

b) Des variations de d'_{AB} on déduit son signe et donc le tableau de variation de d_{AB} :

x	$-\infty$		x_0		$+\infty$
$d'_{AB}(x)$		$-$	0	$+$	
$d_{AB}(x)$	$+\infty$		m		$+\infty$

c) On a, par définition de x_0 : $\dfrac{x_0}{\sqrt{x_0^2 + {y_A}^2}} + \dfrac{x_0 - x_B}{\sqrt{(x_0 - x_B)^2 + {y_B}^2}} = 0.$



Désignons par α et β des mesures respectives des angles $(\overrightarrow{IO},\overrightarrow{IA})$ et $(\overrightarrow{IH},\overrightarrow{IB})$.

Le cosinus d'un angle orienté de vecteurs étant égal au cosinus de l'angle géométrique formé par ces vecteurs, on constate que :

$$\cos\alpha = \frac{x_0}{\sqrt{x_0^2 + {y_A}^2}} \quad \text{et} \quad \cos\beta = \frac{x_0 - x_B}{\sqrt{(x_B - x_0)^2 + {y_B}^2}}.$$

La relation qui définit x_0 se traduit donc par :

$$\cos\alpha = \cos\beta.$$

D'autre part, puisque $(\overrightarrow{IO},\overrightarrow{IA}) = (\overrightarrow{IO},\overrightarrow{IH}) + (\overrightarrow{IH},\overrightarrow{IA}) \quad [2\pi]$, si on désigne par γ une mesure de l'angle $(\overrightarrow{IH},\overrightarrow{IA})$, on a la relation : $\alpha = \pi + \gamma \quad [2\pi]$, qui nous donne : $\sin\alpha = -\sin\gamma$.

Or $\sin\gamma = \dfrac{\overline{OA}}{\|\overrightarrow{IA}\|}$, et $\overline{OA} = y_A > 0$, donc $\sin\gamma > 0$ et donc $\sin\alpha < 0$.

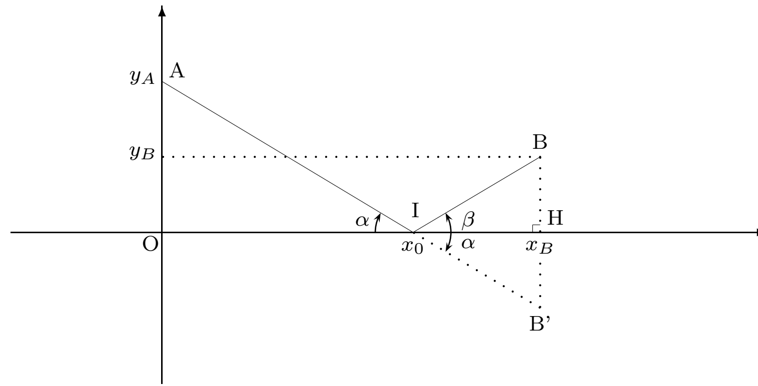
De même, $\sin\beta = \dfrac{\overline{HB}}{\|\overrightarrow{IB}\|}$, et $\overline{HB} = y_B > 0$, donc $\sin\beta > 0$.

On en déduit :

$$\sin\alpha = -\sin\beta.$$

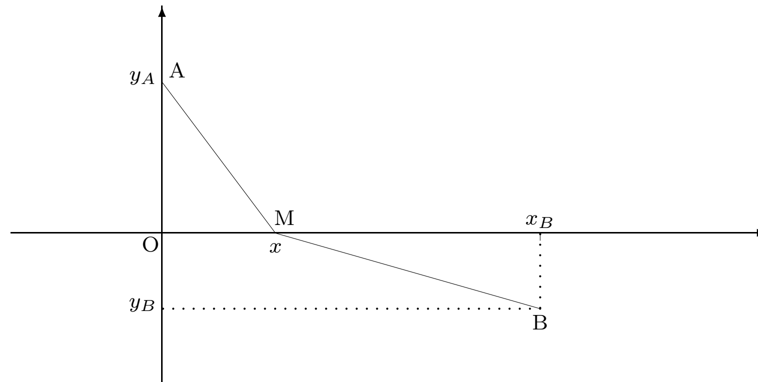
D'où, $\alpha = -\beta \quad [2\pi]$, et on obtient finalement l'égalité :

$$(\overrightarrow{IO},\overrightarrow{IA}) = -(\overrightarrow{IH},\overrightarrow{IB}).$$



Remarque : Si on désigne par B' le symétrique du point B par rapport à l'axe (Ox), une mesure de l'angle $(\overrightarrow{IH}, \overrightarrow{IB'})$ étant α, on en déduit que les points A, I et B' sont alignés, ce qui nous permet d'obtenir très simplement le point B'.

Exercice 4.



a) Le temps $T_{AB}(x)$ mis par le mobile pour aller de A en B en passant par M est donné par :

$$T_{AB}(x) = \frac{\sqrt{x^2 + {y_A}^2}}{v_1} + \frac{\sqrt{(x - x_B)^2 + {y_B}^2}}{v_2}.$$

b) y_A et y_B n'étant pas nuls, les deux radicandes ne sont jamais nuls, et donc la fonction d_{AB} est **dérivable sur** $\mathbb{R}$; et on obtient, pour tout x réel :

$$T'_{AB}(x) = \frac{x}{v_1\sqrt{x^2 + {y_A}^2}} + \frac{x - x_B}{v_2\sqrt{(x - x_B)^2 + {y_B}^2}}.$$

On remarque que la fonction T'_{AB} est la somme de deux fonctions **strictement croissantes** sur $\mathbb{R}$. En effet, si on calcule les dérivées de ces fonctions, on va constater, comme on l'a déjà fait dans l'exercice **2.** (question **a**), que ce sont des réels > 0.

On en déduit que la fonction T'_{AB} est **strictement croissante** sur $\mathbb{R}$.

D'autre part, x_B étant > 0, on a $T'_{AB}(0) < 0$ et $T'_{AB}(x_B) > 0$.

La fonction T'_{AB} étant **continue**, il découle du corollaire du théorème des valeurs intermédiaires que, dans l'intervalle $[0\,, x_B]$, l'équation $T'_{AB}(x) = 0$ possède une solution unique x_0.

De plus, du fait de la **monotonie** de T'_{AB}, si $x < 0$ alors $T'_{AB}(x) \leq T'_{AB}(0)$, donc $T'_{AB}(x) < 0$, et si $x > x_B$ alors $T'_{AB}(x) \geq T'_{AB}(x_B)$, donc $T'_{AB}(x) > 0$; ce qui montre qu'en dehors de l'intervalle $[0\,, x_B]$, l'équation $T'_{AB}(x) = 0$ ne possède aucune solution.

En résumé, la fonction T'_{AB} s'annule en un point x_0 et un seul, compris entre 0 et x_B.

On en déduit le tableau de variation de la fonction T'_{AB} :

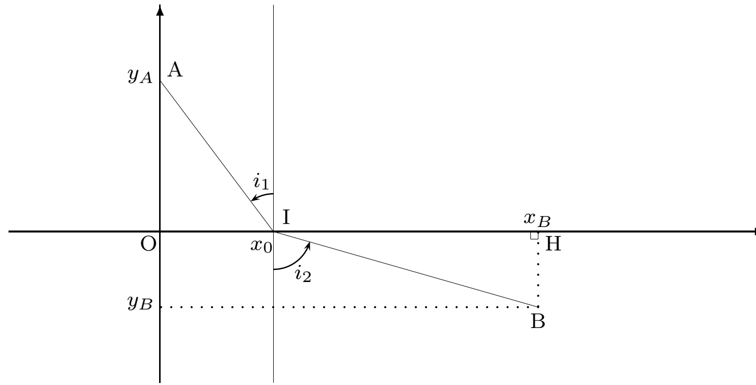
x	$-\infty$	x_0	$+\infty$
$T'_{AB}(x)$	↗	0	↗

c) Des variations de T'_{AB} on déduit son signe et donc le tableau de variation de T_{AB} :

x	$-\infty$	x_0	$+\infty$
$T'_{AB}(x)$	$-$	0	$+$
$T_{AB}(x)$	$+\infty$ ↘	m	↗ $+\infty$

d) On a, par définition de x_0 : $\dfrac{x_0}{v_1\sqrt{x_0^2+{y_A}^2}}+\dfrac{x_0-x_B}{v_2\sqrt{(x_0-x_B)^2+{y_B}^2}}=0.$

Désignons par i_1 et i_2 des mesures respectives des angles $(\overrightarrow{OA},\overrightarrow{IA})$ et $(\overrightarrow{AO},\overrightarrow{IB})$.



On a, d'une part, $\cos(\dfrac{\pi}{2}+i_1)=\dfrac{\overline{IO}}{\|\overrightarrow{IA}\|}=\dfrac{-x_0}{\sqrt{x_0^2+{y_A}^2}}.$

D'où, puisque $\cos(\dfrac{\pi}{2}+i_1)=-\sin i_1$, $\quad\sin i_1=\dfrac{x_0}{\sqrt{x_0^2+{y_A}^2}}.$

D'autre part, $\cos(-\dfrac{\pi}{2}+i_2)=\dfrac{\overline{IH}}{\|\overrightarrow{IB}\|}=\dfrac{x_B-x_0}{\sqrt{(x_B-x_0)^2+{y_B}^2}}.$

D'où, puisque $\cos(-\dfrac{\pi}{2}+i_2)=\sin i_2$, $\quad\sin i_2=\dfrac{x_B-x_0}{\sqrt{(x_B-x_0)^2+{y_B}^2}}.$

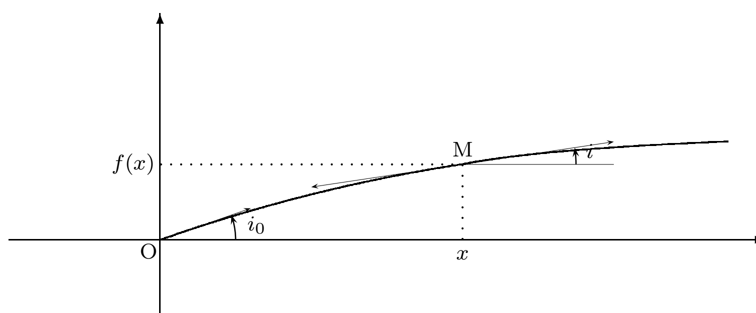
La relation qui définit x_0 se traduit donc par :

$$\frac{\sin i_1}{v_1} - \frac{\sin i_2}{v_2} = 0,$$

et on obtient finalement :

$$\frac{1}{v_1} sin(\overrightarrow{OA}, \overrightarrow{IA}) = \frac{1}{v_2} sin(\overrightarrow{AO}, \overrightarrow{IB}).$$

Exercice 5.



Préliminaire : Montrons que i est une fonction décroissante de x.

Remarquons d'abord que, puisque $\sin(i(x)) = \dfrac{C}{n(x)}$, avec $C > 0$ et n fonction croissante et > 0 de x, la fonction $\sin(i)$ est une fonction décroissante de x.

Considérons à présent deux réels x_1 et x_2 tels que $0 \leq x_1 \leq x_2$ et supposons que $i(x_1) < i(x_2)$.

Comme $i(x_1)$ et $i(x_2)$ sont dans l'intervalle $]0\,;\frac{\pi}{2}[$, et que la fonction sinus est strictement croissante sur cet intervalle, on en déduit que $\sin(i(x_1)) < \sin(i(x_2))$, ce qui est en contradiction avec la décroissance de la fonction $\sin(i)$. Donc $i(x_1) \geq i(x_2)$, et la fonction i est bien une fonction décroissante de x.

Il en découle en particulier que $i \leq i_0$.

Expression de $f'(x)$ lorsque i_0 est petit.

On suppose que l'angle i_0 est suffisamment petit pour que, lorsque $0 \leq i \leq i_0$, on puisse avoir $\tan(i) \approx \sin(i)$. Soit M le point de coordonnées $(x, f(x))$. La tangente en M a pour coefficient directeur $\tan i(x)$, si bien qu'on peut écrire :

$$f'(x) = \tan(i(x)) \approx \sin(i(x)) = \frac{C}{n(x)}.$$

5.A a) Cas où $n = n_0(1+kx),\ k > 0$.

$x > 0$ étant fixé, on a pour tout u compris entre 0 et x :

$$f'(u) = \frac{C}{n_0(1+ku)} = \frac{i_0}{1+ku},$$

puisque $\dfrac{C}{n_0} = \sin(i_0) \approx i_0$.

Cette fonction étant continue en u on peut intégrer de 0 à x :

$$f(x) - f(0) = i_0 \int_0^x \frac{1}{1+ku}\, du,$$

i.e. puisque $f(0) = 0$,

$$f(x) = \frac{i_0}{k} \ln(1+kx).$$

b) Cas où $n = n_0(2 - e^{-\alpha x}),\ \alpha > 0$.

$x > 0$ étant fixé, on a comme précédemment, pour tout u compris entre 0 et x :

$$f'(u) = \frac{C}{n_0(2-e^{-\alpha u})} = \frac{i_0}{2-e^{-\alpha u}}.$$

D'où, en intégrant de 0 à x :

$$f(x) = i_0 \int_0^x \frac{du}{2-e^{-\alpha u}} = i_0 \int_0^x \frac{e^{\alpha u}}{2e^{\alpha u}-1}\, du = \frac{i_0}{2\alpha} \ln(2e^{\alpha x} - 1).$$

c) Cas où $n = \dfrac{n_0(\alpha x+1)}{x+1},\ \alpha > 1$.

$x > 0$ étant fixé, on a comme précédemment, pour tout u compris entre 0 et x :

$$f'(u) = \frac{C(u+1)}{n_0(\alpha u+1)} = \frac{i_0(u+1)}{\alpha u+1}.$$

Or,

$$\frac{u+1}{\alpha u+1} = \frac{\frac{1}{\alpha}(\alpha u+1) - \frac{1}{\alpha} + 1}{\alpha u+1} = \frac{1}{\alpha} + \frac{1-\frac{1}{\alpha}}{\alpha u+1}.$$

On peut donc intégrer $f'(u)$ de 0 à x, ce qui donne finalement :

$$f(x) = i_0 \left[\frac{1}{\alpha}x + \left(1-\frac{1}{\alpha}\right)\frac{1}{\alpha}\ln(\alpha x+1)\right].$$

5.B a)1. On sait que : $sin(i(x)) = \frac{C}{n(x)}$, et que $0 < i < \frac{\pi}{2}$, donc :

$$f_e'(x) = \tan(i(x)) = \frac{sin(i(x))}{\sqrt{1-\sin^2(i(x))}} = \frac{\frac{C}{n(x)}}{\sqrt{1-\frac{C^2}{n^2(x)}}}.$$

D'où, en utilisant la positivité de n :

$$f_e'(x) = \frac{C}{\sqrt{n^2(x)-C^2}}.$$

2. Comme $n(x) = n_0\sqrt{ax+1}$, il vient : $f_e'(x) = \frac{C}{\sqrt{n_0^2(ax+1)-C^2}}$.

On intègre :

$$f_e(x) - f_e(0) = \frac{C}{n_0}\int_0^x \frac{du}{\sqrt{au+1-\left(\frac{C}{n_0}\right)^2}}.$$

D'où, finalement, puisque $f_e(0) = 0$,

$$f_e(x) = \frac{2C}{a\,n_0}\left(\sqrt{ax+1-\frac{C^2}{n_0^2}} - \sqrt{1-\frac{C^2}{n_0^2}}\right).$$

b)1. On sait que, puisque $0 < i < \frac{\pi}{2}$,

$$f_e'(x) = \tan(i(x)) \approx \sin(i(x)) = \frac{C}{n(x)},$$

d'où :

$$f_a'(x) = \frac{C}{n_0\sqrt{ax+1}}.$$

2. On intègre :

$$f_a(x) - f_a(0) = \frac{C}{n_0}\int_0^x \frac{du}{\sqrt{au+1}} = \frac{2C}{a\,n_0}\left(\sqrt{ax+1} - 1\right).$$

D'où, finalement, puisque $f_a(0) = 0$,

$$f_a(x) = \frac{2C}{a\,n_0}\left(\sqrt{ax+1} - 1\right).$$

c)1. La fonction Δ est dérivable sur $\mathbb{R}_+$ comme différence de fonctions dérivables sur $\mathbb{R}_+$, et

$$\Delta'(x) = f_e'(x) - f_a'(x) = \frac{C}{\sqrt{n_0^2(ax+1) - C^2}} - \frac{C}{n_0\sqrt{ax+1}}.$$

D'où

$$\Delta'(x) = \frac{C}{n_0\sqrt{ax+1-\frac{C^2}{n_0^2}}} - \frac{C}{n_0\sqrt{ax+1}}.$$

Or, il est clair que

$$\sqrt{ax+1-\frac{C^2}{n_0^2}} < \sqrt{ax+1},$$

donc

$$\frac{C}{n_0\sqrt{ax+1-\frac{C^2}{n_0^2}}} > \frac{C}{n_0\sqrt{ax+1}},$$

et finalement $\Delta'(x) > 0$ sur $\mathbb{R}_+$, ce qui prouve que la fonction Δ est croissante sur $\mathbb{R}_+$.

c)2. D'après les résultats précédents :

$$\Delta(x) = \frac{2C}{a\,n_0}\left(\sqrt{ax+1-\frac{C^2}{n_0^2}} - \sqrt{1-\frac{C^2}{n_0^2}}\right) - \frac{2C}{a\,n_0}\left(\sqrt{ax+1}-1\right).$$

Donc

$$\Delta(x) = \frac{2C}{a\,n_0}\left(\sqrt{ax+1-\frac{C^2}{n_0^2}} - \sqrt{ax+1} - \sqrt{1-\frac{C^2}{n_0^2}} + 1\right).$$

Or, si on pose $h(x) = \sqrt{ax+1-\frac{C^2}{n_0^2}} - \sqrt{ax+1}$, alors

$$h(x) = \frac{-\frac{C^2}{n_0^2}}{\sqrt{ax+1-\frac{C^2}{n_0^2}} + \sqrt{ax+1}},$$

qui tend manifestement vers 0 lorsque x tend vers $+\infty$.

Finalement, Δ admet une limite réelle en $+\infty$ et on a :

$$\lim_{x\to+\infty} \Delta(x) = 1 - \sqrt{1-\frac{C^2}{n_0^2}},$$

limite qui est manifestement positive.

d)

$$q(x) = \frac{\frac{2C}{a\,n_0}\left(\sqrt{ax+1-\frac{C^2}{n_0^2}} - \sqrt{1-\frac{C^2}{n_0^2}}\right)}{\frac{2C}{a\,n_0}\left(\sqrt{ax+1}-1\right)}.$$

Donc

$$\lim_{x\to\infty} q(x) = 1.$$

5.C Extensions.

a) Calcul exact.

1. $f_e'(x) = \tan(i(x)) = \frac{\sin(i(x))}{\sqrt{1-\sin^2(i(x))}}$

Soit, en remplaçant $\sin(i(x))$ par $\frac{C}{n(x)}$,

$f_e'(x) = \frac{\frac{C}{n(x)}}{\sqrt{1-\frac{C^2}{n^2(x)}}} = \frac{C}{\sqrt{n^2(x)-C^2}}.$

2.
Il suffit alors d'intégrer, et d'utiliser l'hypothèse que f_e s'annule en zéro : f_e est donc la primitive de f_e' qui s'annule en zéro, soit, effectivement :

$$f_e(x) = C\int_0^x \frac{du}{\sqrt{(n(u))^2 - C^2}}.$$

b) Calcul approché.
1. En utilisant l'approximation :$\tan(i(x)) \approx \sin(i(x))$, on obtient : $f_e'(x) \approx \sin(i(x)) \approx \frac{C}{n(x)}$.
On a donc bien : $f_a'(x) = \frac{C}{n(x)}$.

2. De même qu'en **a) 2.**, en intégrant et en tenant compte de l'annulation de f_a en zéro,f_a est la primitive de f_a' qui s'annule en zéro, soit :

$$f_a(x) = C\int_0^x \frac{du}{n(u)}.$$

c) $\Delta(x) = f_e(x) - f_a(x)$.
1. Calculons la dérivée de Δ :
$\Delta'(x) = f_e'(x) - f_a'(x)$.
Remplaçons alors $f_e'(x)$ et $f_a'(x)$ par leur expression vue ci-dessus :
On obtient : $\Delta'(x) = \frac{C}{\sqrt{n^2(x)-C^2}} - \frac{C}{n(x)}$.
Or, $0 < \sqrt{n^2(x) - C^2} < n(x)$.
D'où, en passant aux inverses et en tenant compte du fait que la constante C est strictement positive : $\Delta'(x) = \frac{C}{\sqrt{n^2(x)-C^2}} - \frac{C}{n(x)} > 0$.
Δ' étant alors strictement positive sur $\mathbb{R}_+$, la fonction Δ est strictement croissante sur $\mathbb{R}_+$.
Et comme Δ s'annule en zéro, Δ est strictement positive sur $\mathbb{R}_+^*$.
2. Si l'on définit l'ensemble A des valeurs de Δ sur $\mathbb{R}_+$, d'après la propriété de la borne supérieure dans $\mathbb{R}$, si A est une partie non vide (c'est le cas) et majorée de $\mathbb{R}$ alors elle admet une borne supérieure finie,que l'on notera λ. Tandis que l'on posera $\lambda = +\infty$ dans le cas où A n'est pas majorée. Il suffit alors d'appliquer le théorème de la limite monotone , ce qui permet d'obtenir que Δ admet la limite λ, en $+\infty$. De plus, λ est un réel strictement positif (car Δ est strictement croissante sur $\mathbb{R}_+$ et vaut zéro en zéro), dans le cas où $A = \Delta(\mathbb{R}_+)$ est majorée, et $\lambda = +\infty$, sinon.

d) Pour $x > 0$, $q(x) = \frac{f_e(x)}{f_a(x)}$.

1. Calculons la dérivée de q :

$$q'(x) = \frac{f_e'(x).f_a(x) - f_e(x).f_a'(x)}{f_a^2(x)}$$

Remplaçons :

$$q'(x) = \frac{\frac{C^2}{\sqrt{n^2(x)-C^2}}.\int_0^x \frac{du}{n(u)} \ - \ \frac{C^2}{n(x)}.\int_0^x \frac{du}{\sqrt{(n(u))^2-(C)^2}}}{f_a^2(x)}.$$

La dérivée $q'(x)$ est du signe du numérateur $N(x)$:

$$N(x) = \frac{C^2}{\sqrt{n^2(x) - C^2}}.\int_0^x \frac{du}{n(u)} \ - \ \frac{C^2}{n(x)}.\int_0^x \frac{du}{\sqrt{(n(u))^2 - (C)^2}}$$

Soit, encore :

$$N(x) = \frac{C^2}{n(x)}.\int_0^x \left(\frac{1}{\sqrt{1 - \frac{c^2}{n^2(x)}}} \ - \ \frac{1}{\sqrt{1 - \frac{c^2}{n^2(u)}}} \right) \frac{du}{n(u)}$$

Or, la fonction n est positive et croissante sur $\mathbb{R}_+$, donc, pour tout réel x, lorsque u est compris entre zéro et x,

$$\left(\frac{1}{\sqrt{1 - \frac{c^2}{n^2(x)}}} \ - \ \frac{1}{\sqrt{1 - \frac{c^2}{n^2(u)}}} \right)$$

est négatif. (strictement si u est strictement entre zéro et x, et si n est strictement croissante sur $\mathbb{R}_+$.)
En effet, la fonction qui à u associe $\frac{1}{\sqrt{1-\frac{c^2}{n^2(u)}}}$ a le sens de variations inverse de celui de n, par composée de plusieurs fonctions dont trois strictement décroissantes, et les autres strictement croissantes.
Finalement, on a :

$$\frac{C^2}{n(x)}.\int_0^x \left(\frac{1}{\sqrt{1 - \frac{c^2}{n^2(x)}}} \ - \ \frac{1}{\sqrt{1 - \frac{c^2}{n^2(u)}}} \right) \frac{du}{n(u)} \leq 0$$

,
Soit, $q'(x)$ est négatif, et la fonction q est décroissante sur R_+ (strictement si l'on rajoute l'hypothèse de croissance stricte de n sur $\mathbb{R}_+$).

2. Une nouvelle application du théorème de la limite monotone, à la fonction q cette fois-ci, permet d'obtenir l'existence de la limite μ de q en $+\infty$.
Cette limite, d'après le théorème, est la borne inférieure de l'ensemble des valeurs de q sur $\mathbb{R}_+$.
Or, la décroissance de q et la stricte positivité des fonctions f_e et f_a (strictement croissantes à partir de la valeur zéro prise en $x = 0$), ajouté au fait que, d'après la question **c**), Δ étant strictement positive sur $\mathbb{R}_+^*$, d'après la queston **c)1.**, la fonction q est supérieure ou égale à 1 sur $\mathbb{R}_+^*$;
Finalement : $q(\mathbb{R}_+)$ est inclus dans l'intervalle $I = [1, \gamma]$, où γ désigne la limite de q en 0^+
Et nous verrons à la question **e**) que $\gamma = \frac{1}{\sqrt{1-\left(\frac{C}{n_0}\right)^2}}$, Par suite, μ est, lui aussi dans l'intervalle $I = [1, \frac{1}{\sqrt{1-\left(\frac{C}{n_0}\right)^2}}]$.
En conclusion : q admet en $+\infty$ une limite finie μ comprise entre 1 et $\frac{1}{\sqrt{1-\left(\frac{C}{n_0}\right)^2}}$.

e) Etude au voisinage de zéro : Δ tend vers zéro, lorsque x tend vers zéro. C'est évident, car les deux fonctions f_e et f_a tendent alors vers zéro.
Cherchons maintenant la limite de q en 0^+ :
q se présente sous "forme indéterminée", numérateur et dénominateur tendent vers zéro. Appliquons la règle de l'"hôpital" :
regardons le quotient

$$\frac{f_e'(x)}{f_a'(x)} = \frac{\frac{C}{\sqrt{n^2(x)-C^2}}}{\frac{C}{n(x)}} = \frac{1}{\sqrt{1-\left(\frac{C}{n(x)}\right)^2}}$$

Lorsque x tend vers 0^+, ce quotient tend effectivement vers $\frac{1}{\sqrt{1-\left(\frac{C}{n_0}\right)^2}}$,
Conclusion : on a bien prouvé que, lorsque x tend vers zéro, q tend vers $\frac{1}{\sqrt{1-\left(\frac{C}{n_0}\right)^2}}$.

f) On suppose dans cette question que $n(x) = n_0.e^{\alpha x}$, où α est un réel strictement positif. Calculons les intégrales :

$$f_a(x) = C\int_0^x \frac{du}{n(u)} = \frac{C}{n_0}\int_0^x \frac{du}{e^{\alpha u}}$$

$$f_a(x) = \frac{C}{\alpha.n_0}\left(1 - e^{-\alpha x}\right).$$

La limite A de $f_a(x)$ lorsque x tend vers $+\infty$ est donc : $A = \frac{C}{\alpha.n_0}$.

$$f_e(x) = C.\int_0^x \frac{du}{\sqrt{\left(n(u)\right)^2 - \left(C\right)^2}}.$$

$$f_e(x) = C.\int_0^x \frac{du}{\sqrt{\left(n_0.e^{\alpha.u}\right)^2 - C^2}}.$$

$$f_e(x) = \frac{1}{\alpha}\left(\arctan\left(\frac{\sqrt{n_0^2.e^{2.\alpha x} - C^2}}{C}\right) - \arctan\left(\frac{\sqrt{n_0^2 - C^2}}{C}\right)\right)$$

La limite E de $f_e(x)$ lorsque x tend vers $+\infty$ est donc :
$E = \frac{\pi/2 - \arctan\left(\frac{\sqrt{n_0^2 - C^2}}{C}\right)}{\alpha}$.
Enfin, $\mu = \frac{E}{A} = \frac{\arctan\left(\frac{C}{\sqrt{n_0^2 - C^2}}\right)}{\frac{C}{n_0}}$.
A cet effet,On rappelle que, pour tout réel y strictement positif, on a :
$\arctan(y) + \arctan\left(\frac{1}{y}\right) = \frac{\pi}{2}$
Pour $C = \alpha = 1$, et $n_0 = \sqrt{2}$ on obtient :

$$\mu = \frac{\pi\sqrt{2}}{4}.$$

La limite de delta, elle, vaut :

$$E - A = \arctan\left(\frac{C}{\sqrt{n_0^2 - C^2}}\right) - \frac{C}{n_0}.$$

Pour $C = \alpha = 1$, et $n_0 = \sqrt{2}$, on obtient :

$$\lambda = \frac{\pi}{4} - \frac{1}{\sqrt{2}}.$$

g). On suppose ici que n tend vers $+\infty$, en $+\infty$, Voyons alors si $\mu = 1$: Les fonctions f_e et f_a ont respectivement pour limite en $+\infty$ les intégrales généralisées,convergentes ou divergentes vers $+\infty$:

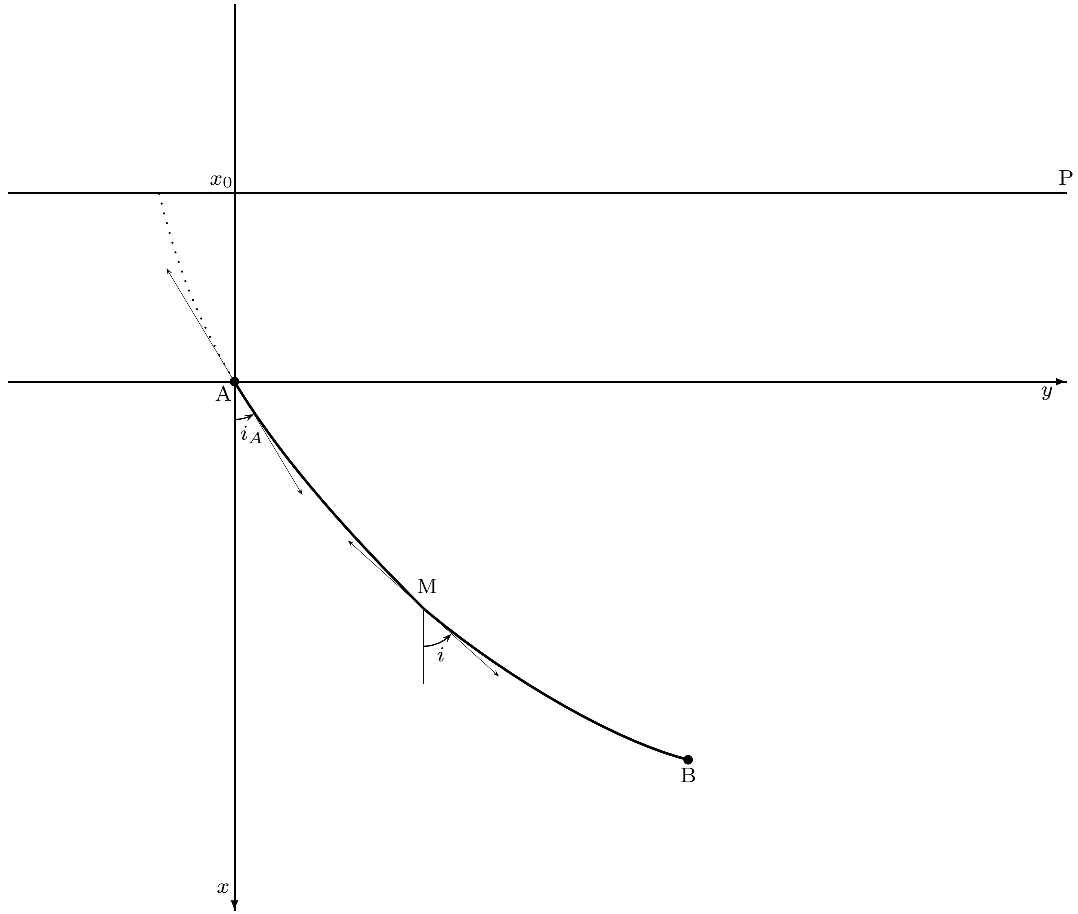
$$E = C.\int_0^{+\infty} \frac{du}{\sqrt{(n(u))^2 - (C)^2}}.$$

$$A = C\int_0^{+\infty} \frac{du}{n(u)}.$$

Les deux fonctions à intégrer étant positives et équivalentes à $\frac{1}{n(u)}$, lorsque u tend vers $+\infty$, les deux intégrales généralisées E et A sont de même nature. En particulier, si , comme au **f)**, $n(u) = n_0.e^{\alpha x}$, pour un réel α strictement supérieur à 1, alors les deux intégrales convergent vers des réels strictement positifs, A et E et $\mu = \frac{E}{A}$
D'après les calculs menés à la question **f)**, il n'est pas obligatoire que $\mu = 1$ puique l'on a trouvé dans le cas éudié en **f)**,
$\mu = \frac{\pi\sqrt{2}}{4}$, différent de 1. Par contre, on avait trouvé $\mu = 1$ au **5.2 d)**.

Les fonctions f_e et f_a ne sont pas nécessairement équivalentes au voisinage de $+\infty$, notamment si μ est différent de 1, ce qui est le cas dans la question **f)**.

Exercice 6.



a) D'après la formule donnée :

$$\sin(i) = \sqrt{x - x_0} \cdot \frac{\sin(i_A)}{\sqrt{|x_0|}},$$

on obtient :

$$\sqrt{x - x_0} = \sqrt{|x_0|} \cdot \frac{\sin(i)}{\sin(i_A)},$$

d'où

$$x = x_0 + |x_0| \cdot \frac{\sin^2(i)}{\sin^2(i_A)} = \phi(i).$$

Remarque : on a $0 < i_A < \pi/2$ d'après le "chapeau" d'introduction.

b) d'après **a)**, $y = f(x) = f(\phi(i))$.

c) La fonction f est supposée dérivable et sa dérivée est donnée par

$$f'(x) = \tan(i(x)),$$

puisque $i(x)$ est l'angle formé par l'axe Ax et le vecteur vitesse au point M de la trajectoire.

La fonction ϕ étant clairement dérivable et f étant supposée dérivable, la composée $f \circ \phi$ est dérivable et

$$(f \circ \phi)'(i) = f'(\phi(i)) \cdot \phi'(i) = f'(x) \cdot \phi'(i).$$

Or

$$\phi'(i) = 2|x_0| \ \frac{\sin(i)\cos(i)}{\sin^2(i_A)},$$

donc, en notant i pour $i(x)$,

$$(f \circ \phi)'(i) = \tan(i) \cdot 2|x_0| \ \frac{\sin(i)\cos(i)}{\sin^2(i_A)} = 2|x_0| \cdot \frac{\sin^2(i)}{\sin^2(i_A)}.$$

d) On a donc

$$y'(i) = \frac{2|x_0|}{\sin^2(i_A)} \cdot \sin^2(i).$$

On intègre y' de i_A à i :

$$y - y_A = \frac{|x_0|}{\sin^2(i_A)} \int_{i_A}^{i} 2\sin^2(u)\,du.$$

Or $y_A = 0$ (car A est l'origine du repère), donc on obtient :

$$y = \frac{|x_0|}{\sin^2(i_A)} \int_{i_A}^{i} 2\sin^2(u)\,du.$$

e) Sachant que

$$2\sin^2(u) = 1 - \cos(2u),$$

on en déduit

$$y = \frac{|x_0|}{\sin^2(i_A)} \int_{i_A}^{i} (1 - \cos(2u))\,du.$$

D'où

$$y = \frac{|x_0|}{\sin^2(i_A)} \left[u - \frac{1}{2}\sin(2u) \right]_{i_A}^{i} = \frac{|x_0|}{\sin^2(i_A)} \left[i - \frac{1}{2}\sin(2i) \right] + y_0,$$

en posant

$$y_0 = -\frac{|x_0|}{\sin^2(i_A)} \left[i_A - \frac{1}{2}\sin(2i_A) \right],$$

puis

$$y = \frac{|x_0|}{2\sin^2(i_A)} [2i - \sin(2i)] + y_0.$$

f) Les coordonnées du point M vérifient donc les relations :

$$\begin{cases} x = \phi(i) = x_0 + |x_0| \cdot \dfrac{\sin^2(i)}{\sin^2(i_A)} \\ \\ y = f(\phi(i)) = y_0 + \dfrac{|x_0|}{2\sin^2(i_A)} [2i - \sin(2i)] . \end{cases}$$

D'où, si on pose

$$K = \frac{|x_0|}{2\sin^2(i_A)},$$

on obtient

$$\begin{cases} x = x_0 + K(2\sin^2(i)) \\ y = y_0 + K\,(2i - \sin(2i))\,, \end{cases}$$

et donc, puisque $2\sin^2(i) = 1 - \cos(2i)$,

$$\begin{cases} x - x_0 = K(1 - \cos(2i)) \\ y - y_0 = K\,(2i - \sin(2i))\,. \end{cases}$$

g) À présent, traçons l'arc (AB) de cette courbe pour les valeurs indiquées.
Remarquons que se donner le point B revient à se donner la valeur de l'angle i_A. En effet, si on se fixe x_B, alors y_B est une fonction continue et strictement croissante de i.

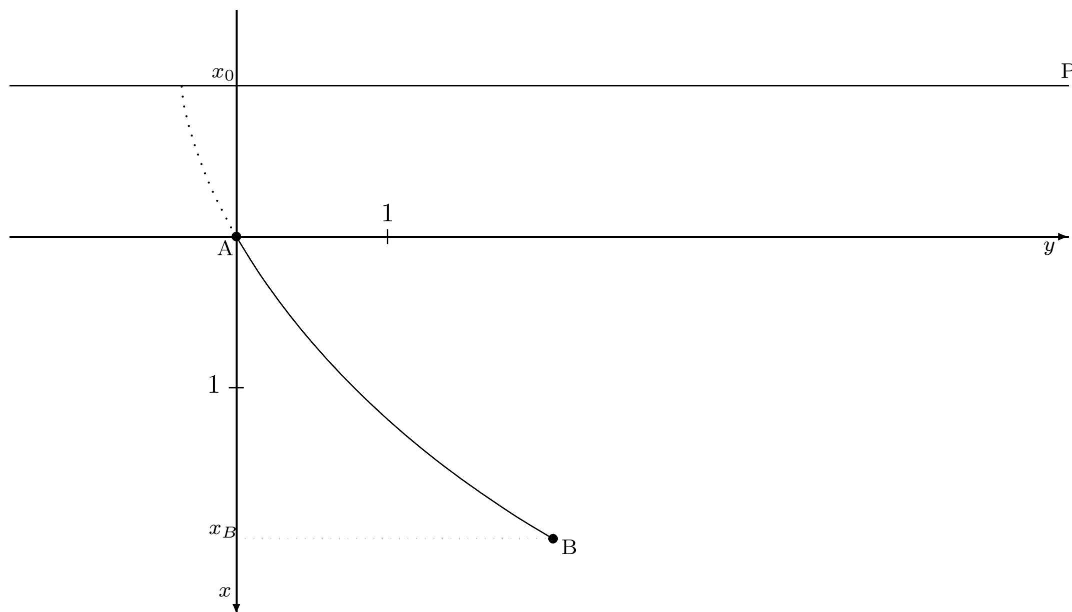
Avec $x_0 = -1$ et $i_A = \dfrac{\pi}{6}$, on trouve $K = 2$ et $y_0 = \sqrt{3} - \dfrac{2\pi}{3}$.

(On peut noter que $y_0 = K[\sin(2i_A) - 2i_A]$.)

Puis, avec $x_B = 2$, on trouve $i_B = \frac{\pi}{3}$.
Ce qui nous donne le système suivant :

$$\begin{cases} x = 2(1 - \cos(2i)) - 1 \\ y = 2\,(2i - \sin(2i)) + \sqrt{3} - \frac{2\pi}{3} \end{cases} \quad , i \in [\frac{\pi}{6}\,;\frac{\pi}{3}].$$

D'où finalement l'arc (AB) :



Exercice 7.

On applique le principe de moindre action : le fil se positionne de manière à minimiser le potentiel U.
Calculons l'élément de potentiel dU correspondant à une longueur élémentaire de fil situé entre l'abscisse x et l'abscisse $x + dx$, de masse élémentaire dm, et à la hauteur y :

$$dU = ydm,$$

où y est une fonction de x à déterminer.
Or, la longeur élémentaire correspondante du fil est :

$$dl = \sqrt{1 + (y\prime(x))^2}.$$

On en déduit :

$$U(x) = \rho \int_0^x y(t) \sqrt{1 + (y\prime(t))^2} dt.$$

On peut assimiler ce problème à la recherche du temps de parcours minimum évoqué au paragraphe **2.**, en remplaçant l'inverse de la vitesse $\frac{1}{v}$ par $\rho.y$, puisque le temps de parcours est le quotient de la distance parcourue avec la vitesse.
On a donc,

$$v = \frac{C}{n} = \frac{1}{\rho y}.$$

Ici, les rôles de x et de y sont échangés par rapport aux notations du paragraphe **2.4.1** , exercice **5.2**.

On obtient :

$$f\prime_e(y) = \frac{\frac{C}{n^2}}{\sqrt{1 - \frac{C^2}{n^2}}}$$

$$f\prime_e(y) = \frac{\frac{1}{\rho y}}{\sqrt{1 - \frac{1}{\rho^2 y^2}}} = \frac{1}{\sqrt{\rho^2 y^2 - 1}}$$

Soit, encore :

$$x = f_e(y) = \int_{y_0}^{y} \frac{dt}{\sqrt{\rho^2 t^2 - 1}}$$

On intègre :
$\frac{1}{\sqrt{\rho^2 t^2 - 1}}$ a pour primitive : $\frac{1}{\rho} Argch\,(\rho t)$, d'où :

$$x = f_e(y) = \frac{1}{\rho} Argch\,(\rho y) - \frac{1}{\rho} Argch\,(\rho y_0)\,.$$

Ajustons les constantes :
Pour $x = a$, on a $y = h$, ce qui donne :

$$\rho.a = Argch(\rho h) - Argch(\rho y_0).$$

D'où finalement :

$$\mathbf{y}(\mathbf{x}) = \frac{\mathbf{1}}{\rho}\mathbf{ch}(\rho\mathbf{x} + \mathbf{b}),$$

où la constante b est égale à :
$b = Argch(\rho y_0) = -\rho a + Argch(\rho h)$.

On obtient bien une chaînette.

Dans le cas particulier où $y_0 = \rho = 1$, alors $b = 0$ et $h = ch(a)$, et l'équation de la courbe devient :

$$\mathbf{y}(\mathbf{x}) = \mathbf{ch}(\mathbf{x}).$$

Deuxième partie

Un peu d'arithmétique algorithmique liée à la cryptographie

8 Introduction

Nous donnons ici des thèmes de projet de travaux pratiques de cryptographie. Bien sûr il ne s'agit pas là de la description du travail à faire, donnée aux étudiants. Il s'agit de l'étude de faisabilité du projet, ainsi que du relevé rapide des outils théoriques et des algorithmes permettant de comprendre le sujet et de le traiter. Il s'agit aussi de prévoir une démarche, qui va des réflexions théoriques jusqu'aux applications concrètes en passant par la mise en place d'algorithmes, leur programmation et les réflexions sur les résultats obtenus. Cette démarche est, nous semble-t-il, apte à éclairer le contexte des outils introduits, et à motiver les débutants dans l'apprentissage du sujet. La programmation et les tests sont faits avec le logiciel xcas [2]. Nous attirons l'attention des enseignants sur le grand intérêt de l'utilisation de ce logiciel dans les classes de divers niveaux.

9 L'algorithme de Hörner

9.1 Présentation

L'algorithme de Hörner est très connu et très simple. Nous en redonnons ici une version plus orientée "informatique". Nous proposons d'autres algorithmes qui l'utilisent d'une manière plus ou moins cachée.

9.2 L'algorithme de Hörner binaire

Soit $a_7a_6a_5a_4a_3a_2a_1a_0$ où les a_i valent 0 ou 1, l'écriture binaire d'un nombre S. Ainsi :

$$S = a_7 2^7 + a_6 2^6 + a_5 2^5 + a_4 2^4 + a_3 2^3 + a_2 2^2 + a_1 2^1 + a_0.$$

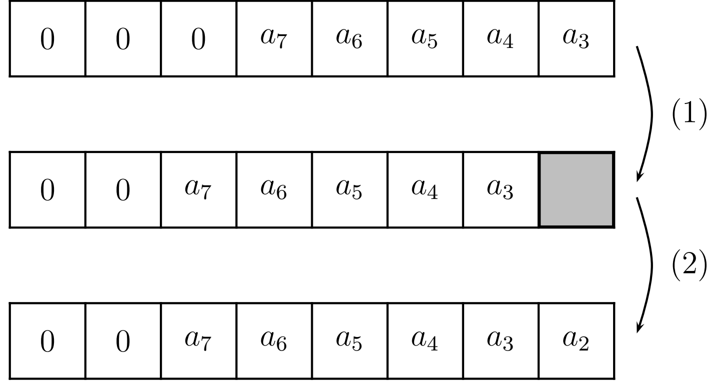


Fig. 1 – La transformation de Hörner binaire

L'algorithme de Hörner peut être vu comme l'entrée en mémoire d'un tel nombre conformément à la figure 1. On utilise les deux opérations :

(1) on décale les bits déjà entrés d'une position vers la gauche,

(2) on insère le bit suivant à la position la plus à droite, laissée libre par le décalage.

Notons S_i le nombre déjà entré, qui s'écrit en binaire $a_7 a_6 \cdots a_{7-i}$. Alors le nombre S_{i+1} obtenu à partir de S_i par transformation de Hörner s'écrit :

$$S_{i+1} = 2S_i + a_{7-i-1}.$$

Dans cette formule la multiplication par 2 correspond au décalage à gauche des bits déjà entrés, et l'addition de a_{7-i-1} correspond à l'insertion du bit suivant à la place laissée libre. Si on décrit la suite des opérations on calcule successivement :

$$S_0 = a_7, S_1 = 2S_0 + a_6, \cdots, S_7 = 2S_6 + a_0,$$

ou encore :

$$S = S_7 = 2(2(2(2(2(2(2a_7 + a_6) + a_5) + a_4) + a_3) + a_2) + a_1) + a_0.$$

L'algorithme est décrit à la figure 2.

Cet algorithme effectue N tours de boucle, chaque boucle coûtant une multiplication par 2 et une addition.

```
entrées : un tableau A de N bits
sortie : le nombre S = ∑_{i=0}^{N−1} A[i]2^i

début
    S ← 0 ;
    i ← N − 1 ;
    tant que i ≥ 0 faire
        S ← 2 ∗ S + A[i] ;
        i ← i − 1 ;
    fintq
    retourner S ;
fin
```

FIG. 2 – L'algorithme de Hörner binaire

9.3 Généralisation

Cet algorithme s'adapte immédiatement à toute écriture polynomiale de la forme :

$$S(X) = a_{N-1}X^{N-1} + \cdots + a_1 X + a_0.$$

Il suffit en effet d'effectuer la suite d'opérations suivantes :

$$\begin{array}{lcl} S_0(X) & = & a_{N-1}, \\ S_1(X) & = & XS_0(X) + a_{N-2}, \\ \cdots & = & \cdots \\ S_{N-1}(X) & = & XS_{N-2}(X) + a_0. \end{array}$$

L'algorithme de Hörner intervient dans de nombreuses situations :

1. évaluation d'un polynôme en un point,
2. traduction binaire - décimal,
3. division euclidienne,
4. calcul d'une puissance.

En fait cette liste n'est pas limitative. En effet, dans de nombreuses situations une partie de l'algorithme consiste à reconstituer un nombre à partir de ses bits. Dans cette situation l'algorithme de Hörner s'applique.

9.4 Exemples

9.4.1 Un exemple direct

Nous nous proposons d'évaluer un polynôme t en un point x. Cette fonction existe dans les systèmes de calcul, néammoins à titre d'exemple nous la reprogrammons dans le style Maple (logiciel utilisé : xcas).

```
evaluation:=proc(t,x)
  local n,i,s;

  n:=length(t);
  i:=n;
  s:=0;
  while (i>=1)
    do
      s:=x*s+t[i];
      i:=i-1;
    od;
  RETURN(s);
end;
```

9.4.2 La division euclidienne

La division euclidienne d'un entier a par un entier $b \neq 0$ consiste à trouver l'unique couple d'entiers (q, r) où q est le quotient et r le reste vérifiant :

$$a = bq + r,$$

avec :

$$0 \leq r < b.$$

On suppose dans la suite que $a \geq 0$ et $b > 0$. Il est facile de voir qu'on peut toujours se ramener à ce cas.

L'algorithme le plus simple consiste à soustraire autant de fois b de a qu'il est possible, jusqu'à obtenir un reste $< b$. Voici l'algorithme correspondant :

```
division:=proc(a,b)
  local r,q,u;
```

```
  r:=a;
  q:=0;
  while (r>=b)
    do
      r:=r-b;
      q:=q+1;
    od;
  u:=[q,r];
  RETURN(u);
end;
```

Hélas, on se rend vite compte que cet algorithme n'aboutit pas lorsque a est trop grand devant b, ce qui arrive très souvent dans les applications concrètes qui utilisent de grands nombres. Prendre par exemple un nombre a de 1024 bits (donc de l'ordre de 2^{1024}) et un b ayant 128 bits. Quel est, à la louche, le nombre de soustractions à faire ?

Remarque : On remarque que $2^{10} = 1024$ c'est-à-dire à peu près 10^3. Donc 10 digits binaires donnent environ 3 chiffres décimaux. Un nombre de 1024 bits s'écrit avec environ 310 chiffres décimaux.

Nous présentons un algorithme beaucoup plus performant, basé sur l'écriture binaire des nombres, qui consiste à rechercher les digits successifs du quotient par dichotomie, et à reconstruire ce quotient par l'algorithme de Hörner.

```
divisionbin:=proc(a,b);
  local r,q,aux,n,u;

  r:=a;
  q:=0;
  n:=0;
  aux:=b;
  while (aux <=a)
    do
      aux:=2*aux;
      n:=n+1;
    od;
  while (n >0)
    do
      aux:=aux/2;
```

```
      n:=n-1;
      if (r <aux)
        then
          q:=2*q;
        else
          q:=2*q+1;
          r:=r-aux;
      fi;
    od;
  u:=[q,r];
  RETURN(u);
end;
```

Commentaires sur l'algorithme : dans la première boucle on détermine un entier n tel que $2^{n-1}b \leq a < 2^n b$. Dans la mémoire aux on met $2^n b$. On sait donc que le quotient euclidien de a par b vérifie :

$$2^{n-1} \leq \left\lfloor \frac{a}{b} \right\rfloor < 2^n.$$

En conséquence ce quotient a n digits. Dans la deuxième boucle on lance une dichotomie ($aux := aux/2$) afin de déterminer ces n digits en commençant par ceux de poids fort (à chaque tour le digit trouvé est 0 si $r < aux$ et 1 sinon). On reconstitue alors parallèlement par l'algorithme de Hörner le quotient q (si le digit trouvé est 0 on décale q en faisant $q := 2 * q$, si c'est 1 on prend $q := 2 * q + 1$). On voit que quel que soit b, le nombre de tours de chacune des deux boucles est majoré par le nombre de bits de a (c'est-à-dire la taille de a). Chaque tour de boucle n'implique que des opérations très élémentaires et peu coûteuses.

Preuve de l'algorithme : Montrons que les conditions

$$\begin{cases} a = aux * q + r \\ 0 \leq r < aux \end{cases}$$

sont un invariant de boucle. Ces conditions sont bien réalisées à l'état initial. Nous noterons aux', q', r' les valeurs en sortie de aux, q, r. Si en entrée de boucle les conditions précédentes sont remplies alors :

- **1er cas** : $r < aux'$. Dans ce cas $r' = r$, aux est divisé par 2 tandis que q est multiplié par 2. On a donc bien les conditions indiquées en sortie.

- **2e cas** : Si $r \geq aux'$ alors on sait que

$$aux' \leq r \leq aux = 2 * aux'.$$

  On a aussi $aux' = aux/2$, $q' = 2 * q + 1$, $r' = r - aux'$. On a donc bien les conditions requises.

De plus n décroît strictement. Quand $n = 0$ alors aux contient b, q contient le quotient et r contient le reste.

9.4.3 Le calcul d'une puissance

Il s'agit de calculer dans un groupe a^n (ou na si on note l'opération additivement). Notons $n_k n_{k-1} \cdots n_1 n_0$ la décomposition binaire de l'exposant n. Alors en suivant l'algorithme de Hörner pour calculer n on voit que lorsqu'on multiplie par 2 la valeur courante de l'exposant on élève au carré la valeur courante de la puissance, et quand on ajoute 1 à la valeur courante de l'exposant on multiplie la valeur courante de la puissance par a. L'algorithme, qui suit exactement l'algorithme de Hörner de reconstitution de l'exposant est donc le suivant (voir figure 3) :

```
entrées : un tableau N de K + 1 bits représentant n en binaire
sortie : le nombre P = a^n

début
    P ← 1 ;
    i ← K ;
    tant que i ≥ 0 faire
        P ← P^2 ;
        si N[i] == 1
            alors P = P * a ;
        finsi
        i ← i − 1 ;
    fintq
    retourner P ;
fin
```

FIG. 3 – L'algorithme d'élévation à une puissance

Cet algorithme est adapté à une représentation binaire de l'exposant. Ceci va de soi pour une implémentation "hardware" ainsi que pour une implémentation dans un langage où on a facilement accès aux bits des données. Si, en revanche, on utilise un langage pour lequel l'accès aux bits des données n'est pas prévu, il convient alors, de recalculer les bits en fonction de la valeur globale de l'exposant n. Dans ce cas il vaut mieux utiliser l'algorithme suivant (voir figure 4), qui calcule les bits successifs à partir de ceux de poids faible et qui calcule simultanément la valeur de la puissance :

```
entrées : un nombre a et un nombre n
sortie : le nombre R = a^n

début
    A ← a ;
    N ← n ;
    R ← 1 ;
    tant que N > 0 faire
        si N est pair
            alors
                A ← A^2 ;
                N ← N/2 ;
            sinon
                R ← R * A ;
                N ← N − 1 ;
        finsi
    fintq
    retourner R ;
fin
```

FIG. 4 – Autre algorithme d'élévation à une puissance

10 l'Echange de clé de Diffie-Hellman

10.1 Présentation du problème

Nous présentons ici le **principe de l'échange de clé de Diffie-Hellman**. L'échange de clés s'insère dans un protocole complexe. Il s'agit en général d'établir un canal de communication sûr. Pour cela un utilise le plus souvent, un circuit de chiffrement à clé secrète, beaucoup plus rapide pour chiffrer un flux de données. L'échange de la clé secrète se fait, soit avec un chiffrement à clé publique comme par exemple RSA, soit par un échange de type Diffie-Hellman (basé sur la résistance du problème du logarithme discret et ses variantes : problème de Diffie-Hellman, problème décisionnel de Diffie-Hellman). Cette dernière méthode présente l'avantage d'assurer la

sécurité rétroactive, c'est-à-dire qu'en cas de divulgation d'un élément secret ou privé du système, seul l'échange en cours est affecté, mais pas les échanges précédents. Autrement dit un ennemi qui aurait enregistré pendant une certaine période les chiffrés échangés, ne peut pas remonter dans le temps. En revanche elle est très vulnérable à l'attaque de l'homme au milieu. Couplée avec un procédé d'authentification mutuelle des deux parties, elle est très efficace (cf. [1] p. 191).

Le principe est le suivant : On dispose d'un grand nombre premier p et d'un élément primitif α du corps $\mathbb{Z}/p\mathbb{Z}$ (générateur du groupe multiplicatif $(\mathbb{Z}/p\mathbb{Z})^*$). Chacun des deux interlocuteurs A et B choisit un entier inferieur à l'ordre du groupe. Nous noterons n et m ces deux entiers choisis repectivement par A et par B. A transmet à B le nombre $\alpha^n \mod p$, tandis que B transmet à A le nombre $\alpha^m \mod p$. Chacun des deux interlocuteurs est alors en mesure de calculer $L = \alpha^{mn} \mod p$. En général, ce nombre L est, d'une part, trop long pour servir de clé pour un système à clé secrète, d'autre part, certains bits sont faciles à calculer. Si bien qu'on le hache pour obtenir la clé commune $K = h(L)$ (bien entendu cette clé K est reconstruite à chaque nouvelle session avec des nouveaux paramètres n et m) (cf. [1] p. 81).

10.2 Mise en place du système

Nous allons simuler un tel système en utilisant un logiciel de calcul. Le logiciel utilisé est **xcas** [2].

La mise en place du système passe par la construction d'un grand nombre premier p et la détermination d'un élément primitif α. Rappelons (cf. [1]) que pour tester si un élément a est primitif de manière efficace, on ne dispose guère que de l'algorithme qui suit : Supposons que :

$$p - 1 = \prod_{i=1}^{k} q_i^{s_i}$$

soit la décomposition de $p - 1$ en facteurs premiers. Alors pour que a soit primitif il faut et il suffit que les k nombres

$$a^{\frac{p-1}{q_i}} \mod p$$

soient tous différents de 1. Pour réaliser ce test il faut disposer de la factorisation de $p - 1$, ce qui n'est pas le cas si on prend p au hasard. On

va donc commencer par construire un grand nombre premier q, puis chercher un nombre premier p de la forme $2kq + 1$ (n'oublions pas que p sera nécessairement impair), de telle sorte que k soit suffisamment petit pour pouvoir être factorisé. On peut se demander si lorsque q est déterminé, on va pouvoir facilement trouver un p qui convienne. Le théorème de Dirichlet sur la densité des nombres premiers d'une progression arithmétique répond à cette question (cf. [1] p. 366). On doit trouver un tel p en temps moyen de l'ordre $\ln(p)$, c'est-à-dire de la taille prévue pour p.

10.3 Programmation des fonctions utiles

```
##################
## alea(x)
##################

#### entree : nombre de bits x
#### sortie : nombre aleatoire s de x bits exactement

alea:=proc(x)
  local n,s;
  s:=1;
  for n from 1 to x-1
    do
      s:=rand(2)+2*s;
    od;
  RETURN(s);
end;


##################
## nbp(x)
##################

#### entree : nombre de bits x
#### sortie : nombre premier aleatoire p de x bits

nbp:=proc(x)
```

```
  local p,a;
  p:=0;
  while (p=0)
    do
      a:=alea(x);
      p:=nextprime(a);
      if p>=2^x
        then p:=0;
      fi;
    od;
  RETURN(p);
end;


##################
## trouvepqk(x,y)
##################

#### entree : nombre de bits x d'un nombre premier q
              nombre de bits y d'un nombre premier p
              ces deux nombres sont tels que q divise p-1
#### sortie : p,q, k tel que p=2kq+1,
####          compteur est le nombre d'iteratons pour
####          trouver p, une fois q construit


trouvepqk:=proc(x,y)
  local z,k,q,deuxq,p,compteur,u;
  compteur:=1;
  z:=y-x-1;
  q:=nbp(x);
  deuxq:=2*q;
  k:=alea(z);
  p:=deuxq*k+1;
  while (is_pseudoprime(p)=0)
    do
      p:=p+deuxq;
      compteur:=compteur+1;
```

```
      k:=k+1;
    od;
  u:=[p,q,k,compteur];
  RETURN(u);
end;

##################
## prim(p,q,k)
##################

#### entree : p,q,k de la sortie de la procedure precedente
####          p=2kq+1
#### sortie : Un element primitif alpha de Z/pZ

prim:=proc(p,q,k)
  local f,j,deuxk,t,alpha,y;
  deuxk:=2*k;
  f:=ifactors(deuxk);
  t:=size(f);
  y:=0;
  alpha:=1;
  while (y=0)
    do
      y:=1;
      alpha:=alpha+1;
      if powmod(alpha,deuxk,p)=1
        then
          y:=0;
        else
          j:=1;
          while ((y=1) and (j <t))
            do
              if powmod(alpha,iquo(p-1,f[j]),p)=1
                then y:=0;
              fi;
              j:=j+2;
            od;
      fi;
```

```
    od;
    RETURN(alpha);
end;
```

10.4 Une session de calcul

Voici une session de calcul avec xcas. On choisit pour q une taille de 874 bits, et pour p une taille de 1024 bits. La taille de $2 * k$ doit donc être de 150 bits, et celle de k de 149 bits, ce qui fait que k peut être facilement factorisé.

On cherche p, q et k tels que $p = 2kq + 1$.

$$u := \mathrm{of}(trouvepqk, [874, 1024]) =$$

$$[84743470264792061813232863946320650474678216
0
615330064498702402030077446271801843972851 91
190694228270005581829053369227212731793953 95
241362667397405709921121175239324973867161 23
701553808210919829676316034504078440181209 05
641325076878185419796940300681006856698631 37
517754071186978936314033716854368882800052 1,$$

$$108573171762429203892430147069079011752799 77
968681658185247261052686626266271062419845 79
914558238821717748754681820254479275602609 86
853875270481845933424863509188527551825575 87
713441412228519547664662390968531329497454 83
220243125413211245351441347395621829307504 67,$$

$$390259715587109698032584610171005489694708780,$$

$$490]$$

On remarque qu'on a fait 490 itérations avant de trouver p à partir de q.

On extrait du tableau les paramètres qui vont servir par la suite :

$$p := \mathrm{at}(u, 0) =$$

847434702647920618132328639463206504746782160
61533006449870240203007744627180184397285191
19069422827000558182905336922721273179395395
24136266739740570992112117523932497386716123
70155380821091982967631603450407844018120905
64132507687818541979694030068100685669863137
5177540711869789363140337168543688828000521

$$q := \mathrm{at}(u, 1) =$$

10857317176242920389243014706907901175279977
96868165818524726105268662626627106241984579
91455823882171774875468182025447927560260986
85387527048184593342486350918852755182557587
71344141222851954766466239096853132949745483
22024312541321124535144134739562182930750467

$$k := \mathrm{at}(u, 2) =$$

390259715587109698032584610171005489694708780

La fonction prim() calcule la factorisation de k. On dispose alors à ce stade de la factorisation complète de $p - 1$. La fonction prim() calcule alors un élément primitif α :

$$alpha := \mathrm{of}(prim, [p, q, k]) = 58.$$

Le système est mis en place.

Remarque : Dans certaines situations il n'est pas souhaitable que l'élément primitif trouvé soit un (petit) diviseur de $p-1$. On a souvent intérêt à partir d'une valeur tirée au sort et non pas de 2 comme on a fait ici (cf. [1] p. 68).

11 La signature avec ombre des cartes bancaires

11.1 Présentation du problème

Nous présentons ici le **principe de la signature avec ombre** utilisée pour les cartes bancaires. Les nombres que nous construisons pseudo-aléatoirement sont évidemment différents de ceux utilisés pour la carte bancaire et d'ailleurs plus longs. La signature avec ombre **n'est que l'un des systèmes de sécurité** des cartes bancaires. Il s'applique dans le cas (de plus en plus rare) où le terminal de paiement du commerçant n'est pas connecté au réseau des banques. Dans ce cas il faut vérifier une **"cohérence interne"** de la carte pour garantir sa conformité.

Le principe est le suivant : Un système de signature RSA est mis en place. Le module n est le produit de deux grands nombres premiers p et q (gardés secrets par les concepteurs) de telle sorte que $\phi(n) = (p-1)(q-1)$ ne soit pas divisible par 3. Cette dernière condition permet de travailler avec l' exposant public $e = 3$. L'exposant privé d (qui n'est bien entendu connu que des concepteurs) vérifie $ed \equiv 1 \quad (\phi(n))$. À chaque carte est associé un nombre I de k bits à partir duquel est construit le nombre "ombré" J de $2 * k$ bits obtenu en concaténant I avec lui-même. Autrement dit, $J = I * 2^k + I$. Ce nombre J est signé avec la clé privée d du système : les concepteurs calculent le nombre $A = J^d \mod n$. Les deux nombres I et A sont stockés dans la carte, dans une zone accessible en lecture, après déblocage par l'entrée du bon PIN-code. La vérification se fait alors de la manière suivante : Après l'entrée du bon PIN-code par le possesseur de la carte, le terminal bancaire récupère I (et donc J) et A. Il calcule $A^3 \mod n$ et vérifie qu'il trouve bien J.

Que peut faire un pirate qui veut créer une carte valide vis à vis de cette protection ? Le module n est bien entendu public. Les concepteurs n'ont pas de raison de le divulguer, mais dans ce cas il peut être facilement retrouvé, soit par le calcul (comme on va le voir dans la suite), soit par examen du programme d'un terminal de paiement. Si le pirate ne sait pas factoriser n (ce qui devrait être le cas pourvu que n soit choisi assez grand), il peut penser à partir à l'envers : Il choisit un A, calcule $J = A^3 \mod n$. La probabilité pour qu'il tombe sur un nombre J "ombré" est infime.

Ce thème est inspiré d'une affaire célèbre de cassage de la carte bancaire

(cf. [1] p. 194). Le module était alors trop petit et a pu être factorisé (après avoir été retrouvé par désassemblage de la mémoire morte d'un terminal de paiement).

11.2 Mise en place du système

Nous allons simuler un tel système en utilisant un logiciel de calcul. Le logiciel utilisé est **xcas** [2]

11.2.1 Le module

Nous allons construire deux nombres premiers p et q de 512 bits chacun au (pseudo) hasard. Pour construire p on construit au hasard un nombre a de 512 bits et on utilise la fonction **nextprime()** appliquée à a. Pour q on refait la même chose à partir d'un autre nombre b au hasard, de 512 bits. (Pour les principes qui soutiennent la fonction nextprime(), se référer au cours [1] p. 365 (densité des nombres premiers, tests de primalité)). On construira p et q tels que $(p-1)$ et $(q-1)$ ne soient pas divisibles par 3.

11.2.2 Les exposants de chiffrement et de déchiffrement

Les deux nombres p et q étant construits, la valeur de e ayant été fixée à 3, il ne reste plus qu'à calculer d. Pour cela il faut résoudre l'équation de Bézout :

$$ed - k(p-1)(q-1) = 1.$$

La méthode standard est l'algorithme d'Euclide étendu (cf. [1] p. 343). Cependant ici, comme $e = 3$, on sait qu'il existe une solution avec $1 \leq k \leq 2$. Il suffit donc de tester lequel des deux nombres $1+(p-1)(q-1)$, $1+2(p-1)(q-1)$ est divisible par 3. On en déduit tout de suite d.

11.2.3 Programmation des fonctions utiles

```
################
## alea(x)
################

#### entree : nombre de bits x
```

```
#### sortie : nombre aleatoire s de x bits (exactement)
#### Algorithme de type Horner

alea:=proc(x)
  local n,s;
  s:=1;
  for n from 1 to x-1
    do
      s:=rand(2)+2*s;
    od;
  RETURN(s);
end;

################
## nbp(x)
################

#### entree : nombre de bits x
#### sortie : nombre premier aleatoire p de x bits (exactement)
#### tel que p-1 ne soit pas divisible par 3

nbp:=proc(x)
  local p,a;
  p:=0;
  while (p=0)
    do
      a:=alea(x);
      p:=nextprime(a);
      while (irem(p-1,3)=0)
        do
          p:=nextprime(p+1);
        od;
      if p>=2^x
        then p:=0;
      fi;
    od;
  RETURN(p);
end;
```

```
################
## RSA(x)
################

#### entree : nombre de bits x de p et de q
#### sortie : u:=[p,q,n,f,ee,d]
####     p et q sont deux nombres premiers aléatoires de x bits
####     n est le produit pq , f est le produit (p-1)(q-1)
####     non divisible par 3,
####     ee est l'exposant public 3, d est l'exposant prive.

rsa:=proc(x)
  local u,p,q,n,f,ee,d;
  p:=nbp(x);
  q:=nbp(x);
  n:=p*q;
  f:=(p-1)*(q-1);
  ee:=3;
  if irem(1+f,3)=0
    then d:=iquo(1+f,3);
    else d:=iquo(1+2*f,3);
  fi;
  u:=[p,q,n,f,ee,d];
  RETURN(u);
end;

################
## J(x)
################

#### entree : nombre de bits x
#### sortie : m un nombre aleatoire avec ombre, de 2x bits.

J:=proc(x)
  local m;
  m:=alea(x);
  m:=2^x *m+m;
```

```
  RETURN(m);
end;
```

11.2.4 Une session de calcul

On a reproduit ici les étapes importantes de la session.

Calcul d'un nombre m ombré.

$$m := \mathrm{of}(J, [160]) =$$

1913812128608185951730736219776809437753711184l2

884555674940505693432831788645316196006997452301 5

Vérification que m est bien ombré.

$$\mathrm{iquo}(m, 2^{160}) =$$

1309483396887138809521229406871667794774305602695

$$\mathrm{irem}(m, 2^{160}) =$$

1309483396887138809521229406871667794774305602695

Mise en place du système de signature RSA.

$$u := \mathrm{of}(rsa, [512]) = [p, q, n, f, ee, d] =$$

[2230990615533910556174725817274793421183869013963680 8

92502859361436191117444048956818641621012746859349461 7

78890377979011890602869274118155684419856272601,

1300702788989767847857136262488132324878521417206394l3

662376116861098013460789493376854188817989665836373072

950472161425866592183276275906315180648380l287,

2901855715834956349085562850185367157341981182764822 51

1077233389215653612700199245281005720491411779628500 4774
2357900151139130770000709033836295792374679162777535 2484
0772847985507568077604035961756191816410917740444732 4948
3723697099503378743410229685091456465932218826307419 5040
470643622228737804453866374 87,

2901855715834956349085562850185367157341981182764822 5110

7723338921565361270019924528100572049141177962850047 7423
5790015113913077000070903383629579237467880960819479 6039
8880766688677044834657533785188011566062647423943202 3124
7164515561279155990103704157374327342382372671414894 7522
362325345654944219046563600,

3,

1934570477223304232723708566790244771561320788509881 6740

5148892614376907513346616352067048032760785308566698 4949
0526676742608718000047268922419719491645253973879653 0693
2587177792451363223105022523458674377375098282628801 5416
4776343707519437326735802771582884894921581780943263 1681
57488356376996281269770 9067]

Après récupération des nombres n et d dans le tableau précédent on calcule la signature de m.

$$a := \text{powmod}(m, d, n) =$$

$$22899095928567191928917700774422548400530644861426055837$$
$$91415063967183317929221785364043661268805336323205741287$$
$$85501809876060102030784245050803262581234568091205163480$$
$$28756394645799810987780067786365460607694481804303264625$$
$$67162506921693213981737812210544435878915258231972719479$$
$$30040049977555388729059039$$

On vérifie la signature.

$$b := \text{powmod}(a, 3, n) =$$

$$19138121286081859517307362197768094377537111841288455567$$
$$49405056934328317886453161960069974523015$$

$$m =$$

$$19138121286081859517307362197768094377537111841288455567$$
$$49405056934328317886453161960069974523015$$

On vérifie que m = b.

11.3 Détermination du module

11.3.1 Principe

Le module, quoique public dans ce système, n'a pas besoin d'être fourni aux utilisateurs. Mais dans ce cas il est facile de le retrouver par observation de deux transactions avec deux cartes distinctes. En effet si on connaît deux couples (J_1, A_1), (J_2, A_2) on peut écrire que :

$$A_1^3 - J_1 = k_1 n,$$

$$A_2^3 - J_2 = k_2 n.$$

En conséquence,

$$\gcd(A_1^3 - J_1, A_2^3 - J_2) = n \gcd(k_1, k_2).$$

Mais la probabilité pour que les nombres k_1, k_2, qu'on peut supposer pris au hasard, aient un gcd supérieur à s est de l'ordre de $1/s^2$. En conséquence on calcule $\gcd(A_1^3 - J_1, A_2^3 - J_2)$, on en cherche les diviseurs < 1000 (par exemple). Après simplification par ces nombres, on obtient très probablement n.

11.3.2 Fonctions utiles

```
##################
## recup(x)
##################

#### entree : deux transaction : J1, A1, J2, A2
#### sortie : le module n


recupn:=proc(J1,A1,J2,A2)
  local a,b,i;
  a:=A1^3-J1;
  b:=A2^3-J2;
  d:=gcd(a,b);
  for i from 2 to 1000
    do
      if irem(d,i)=0
        then d:=iquo(d,i);
      fi;
    od;
  RETURN(d);
end;
```

11.3.3 Un session de calcul

On met en place le système de signature RSA.

$$u := \mathrm{of}(rsa, [512]) = [p, q, n, f, ee, d] =$$

[11557505126668263013248023996772727860861805284452779569587831476635048780969943565287164525899384315131063140185642263415978464285913933254198607716241327,

12987986767676272101949876435216090813643650212656148288042839836346078501707535510192563981752167047649006029335585043227423883557347382786263908859127407,

150108723652518077100249300486683815970940048513403697683409785510894364860275320914369394851882605495302846809417226345174379286694943674357995687755429132702614069402370805692507135146717258729052216625811035983987199056681889790739096064133319873472696389943757487705419172001542296637496548390519051749089,

150108723652518077100249300486683815970940048513403697683409785510894364860275320914369394851882605495302846809417226345174379286694943674357995687755429108157122175057835690494606703157898584223596719516883178353315886075554607113260020584

4048122219213336098745879664781125285991
9445337618050792800247638O356,

3,

1000724824350120514001662003244558773139
6003234226913178893985700726290990685021
3942912929901255070330201897872944817563
4495861911299624495719971251702860721047
4811670522379366307113543859905614906447
9677922118902210590717036404742173347056
2698748146142224065830586443187416857327
963022507870052853349842535711]

En voici les valeurs du module et de l'exposant privé.

$$n := \mathrm{at}(u, 3) =$$

1501087236525180771002493004866838159709
4004851340369768340978551089436486027532
0914369394851882605495302846809417226345
1743792866949436743579956877554291327026
1406940237080569250713514671725872905221
6625811035983987199056681889790739096064
1333198734726963899437574877054191720015
4229663749654839051905174908

$$d := \mathrm{at}(u, 5) =$$

1000724824350120514001662003244558773139

$$600323422691317889398570072629099068502139429129299012550703302018978729448175634495861911299624495719971251702860721047481167052237936630711354385990561490644796779221189022105907170364047421733470562698748146142224065830586443187416857327963022507870052853349842535 71$$

On simule deux cartes (m_1, a_1) et (m_2, a_2) (m_i est la valeur ombrée et a_i la valeur obtenue en signant m_i).

$$m1 := \mathrm{of}(J, [160]) =$$
$$188369833344679268364493657006189812002862245592886882840265056284288920134359410518931374607579 0$$

$$a1 := \mathrm{powmod}(m1, d, n) =$$
$$1782609113328797750877243758176430935878656892428030843556338688078359943055763701294747509043342387367319256826186234349335465629685690953929412180882023806662792793106436289754891114098125545462332071454191250478950393518154149111492384082735385755273414717612631776263957857126887916427340849797539097354 8$$

$$m2 := \mathrm{of}(J, [160]) =$$
$$184024302075389657207809275690704316257 6$$

7394421600930851274760396250107282004677

13820228252686350

$$a2 := \text{powmod}(m2, d, n) =$$

1411747878125505580993699763752515305378

2947154876899570264397187858138418477195

3295947666408330068026135316690352454327

8863136431339736841065121975839408941281

0759543302162131474948270999852785240019

6863866896788451885861844146688711866966

1634609112194492317474446409797700282256

49102925018599070785290361680

$$r := \text{of}(recupn, [m1, a1, m2, a2]) =$$

1501087236525180771002493004866838159709

4004851340369768340978551089436486027532

0914369394851882605495302846809417226345

1743792866949436743579956877554291327026

1406940237080569250713514671725872905221

6625811035983987199056681889790739096064

1333198734726963899437574877054191720015

42296637496548390519051749089

On constate que r=n. On a donc bien récupéré le module.

12 Attaque par faute de la signature RSA

12.1 Présentation du problème

Nous présentons ici le **principe de la signature RSA**. Ce principe de signature peut être utilisé par une carte à puce, qui sert alors à signer. Il faut donc une implémentation efficace du fait des faibles moyens de calcul de la carte. Nous verrons comment accélérer le calcul et, hélas, comment, si on n'y prend garde, fracasser le système par une attaque physico-mathématique (et relativement traumatisante pour la carte).

En pratique la carte à puce dispose en interne (et donc de manière inaccessible) de sa clé privée (clé de signature). Le terminal de vérification (qui a accès aux clés publiques des usagers) envoie au hasard un (ou plusieurs) nombres à signer, la carte fait le calcul en interne dans une zone inaccessible (par un programme non modifiable) de la signature du nombre proposé et la renvoie au terminal vérificateur. Le terminal vérifie avec la clé publique de la carte que la signature est valide.

12.2 Description de la signature RSA

Le principe est le suivant : Un système de signature RSA est mis en place. Le module n est le produit de deux grands nombres premiers p et q. On pose :

$$\phi(n) = (p-1)(q-1).$$

Soit un entier e tel que :

$$1 < e < \phi(n)$$

$$e \wedge \phi(n) = 1.$$

Soit d un entier vérifiant

$$1 < d < \phi(n)$$

$$ed \equiv 1 \quad \phi(n).$$

On définit la clé publique : (n, e) et la clé privée (stockée dans la carte) : d.

Un message $0 < m < n$ est signé en calculant (de manière interne à la carte) :

$$s = m^d \mod n.$$

Le message signé est alors (m, s).

La signature peut être vérifiée (hors de la carte) par tous en calculant :

$$s^e \mod n.$$

Paramètres : $p, q, n, \phi(n), e, d$
Clé publique : (n, e)
Clé privée : d

Signature : (m, s) avec $s = m^d \mod n$
Vérification : $m = s^e \mod n$

12.3 Mise en place effective

12.3.1 Trouver p et q

Les nombres p et q sont obtenus par tirage au sort parmi les nombres impairs de la taille fixée (512 bits) puis testés par l'algorithme de Miller-Rabin (cf. [1] p. 364). Si le test échoue, on incrémente de 2 le nombre qui a été testé non premier et on reteste.

12.3.2 Calculer un e valide

On tire au sort e jusqu'à ce que $\gcd(e, \phi(n)) = 1$.

12.3.3 Calculer d

On utilise l'algorithme d'Euclide étendu appliqué à e et $\phi(n)$ qui nous permet de déterminer l'inverse d de e modulo $\phi(n)$. Cet algorithme est rapide (cf. théorème de Lamé utilisant la suite de Fibonacci).

12.4 Calcul de la signature

12.4.1 Calcul d'une puissance modulo

Il s'agit de calculer $m^d \mod n$. On connaît un algorithme rapide qui permet de faire le calcul. Cet algorithme a été donné dans la section sur l'algorithme de Hörner.

12.4.2 On peut mieux faire (les chinois à la rescousse)

Le propriétaire de la clé privée (la carte) connaît d donc connaît p et q (la connaissance de la factorisation est quasi-équivalente à la connaissance de d. En pratique on peut toujours récupérer p, q à partir de n, e, d). Donc on peut travailler modulo p et modulo q et reconstituer les résultats modulo n par le théorème des restes chinois. Rappelons en outre que grâce au petit théorème de Fermat, travailler modulo p sur des nombres compris entre 1 et $p-1$ permet de travailler modulo $p-1$ sur les exposants affectés à ces nombres.

Posons donc :

$$d_1 = d \mod p-1,$$

$$d_2 = d \mod q-1,$$

$$s_1 = m^{d_1} \mod p,$$

$$s_2 = m^{d_2} \mod q,$$

(c'est plus court à calculer que $m^d \mod n$ car les exposants sont plus courts).

Alors la signature s vérifie :

$$s \equiv s_1 \quad (p),$$

$$s \equiv s_2 \quad (q).$$

Le théorème des restes chinois permet de conclure : on calcule a et b tels que $ap + bq = 1$ et on obtient (vérification immédiate)

$$s = aps_2 + bqs_1 \mod pq.$$

12.5 Une châtaigne bien placée

Supposons que la carte calcule correctement

$$s_1 = m^{d_1} \mod p,$$

mais fasse une erreur dans le deuxième calcul et obtienne

$$s_2' \not\equiv s_2 \quad (q).$$

le s' calculé est
$$s' = aps'_2 + bqs_1 \mod pq.$$
Donc
$$s'^e \equiv m \quad (p),$$
$$s'^e \not\equiv m \quad (q).$$
On conclut que
$$\gcd(s'^e - m, n) = p,$$
et on a factorisé le module uniquement avec des données publiques (et le calcul erroné s' de la vraie signature s).

Il suffit donc de provoquer une erreur de calcul au bon moment pour que la carte "laisse fuir" la factorisation de n. Si on considère que le calcul d'une puissance modulo demande de nombreux cycles du processeur, il est relativement facile d'envoyer une impulsion électromagnétique au circuit au bon moment. Une impulsion trop forte détruit le circuit (on évitera de mettre les cartes à puce dans un four à micro-ondes).

12.6 Programmation des fonctions utiles

```
##################
## alea(x)
##################

#### entree : nombre de bits x
#### sortie : nombre aleatoire s de x bits exactement

alea:=proc(x)
  local n,s;
  s:=1;
  for n from 1 to x-1
    do
      s:=rand(2)+2*s;
    od;
  RETURN(s);
end;
```

```
##################
## toutalea(x)
##################

#### entree : nombre de bits x
#### sortie : nombre aleatoire s de x bits au plus

toutalea:=proc(x)
  local n,s;
  s:=0;
  for n from 1 to x
    do
      s:=rand(2)+2*s;
    od;
  RETURN(s);
end;

##################
## nbpgene(x)
##################

#### entree : nombre de bits x
#### sortie : nombre premier p de x bits exactement

nbpgene:=proc(x)
  local p,a;
  p:=0;
  while (p=0)
    do
      a:=alea(x);
      p:=nextprime(a);
      if p>=2^x
        then p:=0;
      fi;
    od;
  RETURN(p);
end;
```

```
##################
## rsagene(x)
##################

#### entree : nombre de bits x
#### sortie : taleau u=[p,q,n,eul,ee,d]
####          où p et q ont x bits n=pq,
####          eul=(p-1)*(q-1), ee exposant de chiffrement
####          d exposant de déchiffrement (ou de signature)

rsagene:=proc(x)
  local u,p,q,n,eul,ee,d,t;
  p:=nbpgene(x);
  q:=nbpgene(x);
  n:=p*q;
  eul:=(p-1)*(q-1);
  while (ee=0)
    do
      ee:=toutalea(2*x);
      if ((ee>=eul) or (gcd(ee,eul)<>1))
        then ee:=0;
      fi;
    od;
  t:=iabcuv(eul,ee,1);
  d:=t[2];
  if d<0
    then d:=d+eul;
  fi;
  u:=[p,q,n,eul,ee,d];
  RETURN(u);
end;

##################
## siglent(x)
##################

#### entree : message x
```

```
#### sortie : signature du message x avec la clé privée
####          du système stocké dans le tableau u
####          le calcul étant direct.

siglent:=proc(x,u)
  local s;
  s:=powmod(x,u[6],u[3]);
  RETURN(s);
end;

##################
## bezoutpq(u)
##################

#### entree : u tableau contenant la description d'un système RSA
#### sortie : v tableau contenant a et b tels que ap+bq=1

bezoutpq:=proc(u)
  local v;
  v:=iabcuv(u[1],u[2],1);
  RETURN(v);
end;

##################
## sigrap(x)
##################

#### entree : message x, tableau u décrivant le système RSA utilisé,
####          tableau v contenant a et b tels que ap+bq=1.
#### sortie : signature du message x avec la clé privée
####          du système stocké dans le tableau u
####          le calcul étant fait en utilisant le théorème
####          des restes chinois.

sigrap:=proc(x,u,v)
  local s,d1,d2,s1,s2;
  d1:=irem(u[6],u[1]-1);
  d2:=irem(u[6],u[2]-1);
```

```
  s1:=powmod(x,d1,u[1]);
  s2:=powmod(x,d2,u[2]);
  s:=irem(v[1]*u[1]*s2+v[2]*u[2]*s1,u[3]);
  RETURN(s);
end;

##################
## sigraperr(x)
##################

#### entree : message x, tableau u décrivant le système RSA utilisé,
####          tableau v contenant a et b tels que ap+bq=1, un nombre
####          de bits k d'un calcul erroné.
#### sortie : signature erronée du message x avec la clé privée
####          du système stocké dans le tableau u
####          le calcul étant fait en utilisant le théorème
####          des restes chinois à partir d'une valeur erronée de s2.

sigraperr:=proc(x,u,v,k)
  local s,d1,d2,s1,s2;
  d1:=irem(u[6],u[1]-1);
  d2:=irem(u[6],u[2]-1);
  s1:=powmod(x,d1,u[1]);
  s2:=alea(k);
  s:=irem(v[1]*u[1]*s2+v[2]*u[2]*s1,u[3]);
  RETURN(s);
end;

##################
## verif(x,s,w)
##################

#### entree : message x, signature s,
####          tableau w contenant n et ee (clé publique)
#### sortie : 0 ou 1 suivant que la signature est vérifiée ou non.

verif:=proc(x,s,w)
  local s1,r;
```

```
  s1:=powmod(s,w[2],w[1]);
  r:=1;
  if (s1=x)
    then r:=0;
  fi;
  RETURN(r);
end;
```

12.7 Une session de calcul

$$u := rsagene(512) = [p, q, n, eul, ee, d]$$

$$[11557505126668263013248023996772727860861805284452779569587831476635048780969943565287164525899384315131063140185642263415978464285913933254198607716241327,$$

$$129879867676762721019498764352160908136436502126561482880428398363460785017075355101925639817521670476490060293355850432274238835573473827862639088591274 07,$$

$$15010872365251807710024930048668381597094004851340369768340978551089436486027532091436939485188260549530284680941722634517437928669494367435799568775542913270261406940237080569250713514671725872905221662581103598398719905668188979073909606413331987347269638994375748770541917200154229663$$

7496548390519051749089,

1501087236525180771002493004866838159709400485134036976834097855108943648602753209143693948518826054953028468094172263451743792866949436743579956877554291081571221750578356904946067031578985842235967195168831783533158860755546071132600205844048122219213336098745879664781125285991944533761805079280024763803 56,

9483665035412529361663831358835137196006864479231744934885950894332102995714961622903957922238237885020458101562882702333429187781512473493496962580767565707321797623587213666748749799026085483246807770057461146737885059695924072930399026994641246454522694293462333673895274671987779667558666761198972934019 1,

120546943836664743354365345833677169503802695819257585488879650559291133505798980668452031485214484158591515784701993049912782842936147442692853327151389468158530426398358921827160104214946566850008205602010703961882093324570837782536962771914443639733313083138927397586133855154504057 58

5892554631154787116947]

$p := u[1] =$

11557505126668263013248023996772727860861
80528445277956958783147663504878096994356
52871645258993843151310631401856422634159
78464285913933254198607716241327

$q := u[2] =$

12987986767676272101949876435216090813643
65021265614828804283983634607850170753551
01925639817521670476490060293355850432274
23883557347382786263908859127407

$n := u[3] =$

15010872365251807710024930048668381597094
00485134036976834097855108943648602753209
14369394851882605495302846809417226345174
37928669494367435799568775542913270261406
94023708056925071351467172587290522166258
11035983987199056681889790739096064133319
87347269638994375748770541917200154229663
7496548390519051749089

$ee := u[5] =$

94836650354125293616638313588351371960068
64479231744934885950894332102995714961622

9039579222382378850204581015628827023334291877815124734934969625807675657073217976235872136667487497990260854832468077700574611467378850596959240729303990269946412464545226942934623336738952746719877796675586667611989729340191

$$d := u[6] =$$

120546943836664743354365345833677169503802695819257585488879650559291133505798980668452031485214484158591515784701993049912782842936147442692853327151389468158530426398358921827160104214946566850008205602010703961882093324570837782536962771914443639733313083138927397586133855154504057585892554631154787116947

$$w := [n, ee] =$$

[150108723652518077100249300486683815970940048513403697683409785510894364860275320914369394851882605495302846809417226345174379286694943674357995687755429132702614069402370805692507135146717258729052216625811035983987199056681889790739096064133319873472696389943757487705419172001542296637496548390519051749089,

$$9483665035412529361663831358835137196006864479231744934885950894332102995714961622903957922238237885020458101562882702333429187781512473493496962580767565707321797623587213666748749799026085483246807770057461146737885059695924072930399026994641246454522694293462333673895274671987779667558666761198972934 0191]$$

$$v := bezoutpq(u) =$$

$$[-2154803380970768268585732897789405723556024541933935076244633030229555728187104200601054601651271715171040288716263768494054009558618625784480845846194158, 19174758619643599063310871445558540148305419751509253514252584407119243048609748340198172757231502040013924016506373593152267816054628892920289742823 01181]$$

On engendre le message à signer (on prend 128 bits) :

$$m := alea(128) =$$
$$306958758937350771076293918003670230104$$

On signe avec la méthode initiale :

$$slent := siglent(m, u) =$$
$$8314025575800397774593519581246582640 6127$$

52402059191937143136029860476278713114249
28327894197170169173605732905090136534169
83313542482435928055764121891564088976383
99213620278072469465451017430619673245739
89581815342956285474223087001904855193166
92710564766441404507158428251683667587190
844431451897431629163

On signe en utilusant les restes chinois (méthode rapide) et on trouve évidemment la même chose :

$$srap := sigrap(m, u, v) =$$

83140255758003977745935195812465826406127
52402059191937143136029860476278713114249
28327894197170169173605732905090136534169
83313542482435928055764121891564088976383
99213620278072469465451017430619673245739
89581815342956285474223087001904855193166
92710564766441404507158428251683667587190
844431451897431629163

On vérifie la signature :

$$r := verif(m, slent, w) = 0$$

elle est correcte.

On produit une signature par la méthode des reste chinois avec un des deux calculs erroné :

$$serr := sigraperr(m, u, v, 128) =$$

792714385211773136869461638296130737520195

17609998951484970352236662000942336718847041530887732642415348053629883607536207518509309399400194136744528365693741211499542401611916090102768814111064081614459445447189795499237727194373392882133891195051069882869858007127318444223699665834300969766419928731590468

On retrouve alors le facteur p ce qui casse le système :

$z := \text{powmod}(serr, ee, n) =$
119717545057683844592427586350938719992807897142277759144399480741914436634094167869618705054285556667783758643954019153947298767174903283384109163743897164419771733505597197559555187687440096487719258010284167229409963980440336146929743613665592015547654159705451532140298795300137822315827799277859881123798 7

$z1 := z - m =$
119717545057683844592427586350938719992807897142277759144399480741914436634094167869618705054285556667783758643954019153947298767174903283384109163743897164419771733505597197559555187687440096487719258010284167229409963980440336146929743613665592015547654159705451532140268099424244087238720169 8

$$860595141007883$$

$$pexp := \gcd(z1, n) =$$

$$115575051266682630132480239967727278608618$$
$$052844527795695878314766350487809699435652$$
$$871645258993843151310631401856422634159784$$
$$6428591393325419860771624132$$

on vérifie qu'on a bien retrouvé p :

$$differe := pexp - p = 0.$$

13 Attaque de RSA par fraction continue

13.1 Présentation du problème

La primitive de chiffrement RSA (ou de signature) est sensible à une attaque dûe à Wiener, utilisant un développement en fraction continue (voir [1], annexe C, p. 375), dans le cas où l'exposant de chiffrement (ou de signature) d est "petit". Si la taille de d (son nombre de bits) est inférieure au quart de la taille du module, cette attaque fonctionne. Nous allons voir comment la monter.

13.2 Rappels sur la primitive RSA

On rappelle que le système RSA est construit à partir de deux grands nombres premiers distincts p et q dont on note n le produit ($n = pq$). Ce nombre n qui est appelé le module est public (mais bien sûr p et q sont secrets). On note $\phi(n) = (p-1)(q-1)$ (valeur en n de la fonction d'Euler). L'exposant de chiffrement $1 < e < \phi(n)$ est un nombre premier avec $\phi(n)$ qui est aussi public. L'exposant de déchiffrement d (clé privée) est secret. Il est calculé de manière à ce que :

$$ed \equiv 1 \quad (\phi(n)),$$

$$1 < d < \phi(n).$$

Dans la suite on va supposer que $p > \sqrt{n} > q$ et que p et q ont la même taille, ce qui implique que :

$$1 < \frac{p}{q} < 2,$$

On supposera aussi que :

$$\log_2(d) \leq \frac{1}{4}\log_2(n) - 3.$$

13.3 Si on connaît d alors on peut factoriser n

Théorème 13.1 *Il existe un* **algorithme probabiliste** *ayant pour entrées le module n les exposants de chiffrement e et de déchiffrement d, qui calcule la factorisation pq de n.*

Preuve. Remarquons tout d'abord que :

$$x^2 \equiv 1 \quad (n)$$

si et seulement si :

$$\begin{cases} x^2 \equiv 1 \quad (p) \\ \text{et} \\ x^2 \equiv 1 \quad (q). \end{cases}$$

Ces équations sont vérifiées si et seulement si :

$$\begin{cases} x \equiv \pm 1 \quad (p) \\ \text{et} \\ x \equiv \pm 1 \quad (q). \end{cases}$$

L'équation initiale possède quatre solutions, deux d'entre elles sont les solutions triviales vérifiant $x \equiv \pm 1 \quad (n)$.

Si x est une solution non triviale, n divise $(x+1)(x-1)$ mais ne divise ni $(x-1)$ ni $(x+1)$. Donc :

$$\text{pgcd}(x+1, n) = p \text{ ou } q,$$

$$\text{pgcd}(x-1, n) = q \text{ ou } p.$$

Donc, si nous connaissons une racine carrée non triviale de 1 modulo n, nous pouvons calculer en temps polynomial la factorisation de n.

Maintenant le problème qui se pose est : comment calculer une racine carrée non triviale de 1 modulo n ?

Choisissons au hasard w tel que $1 < w \leq n-1$. Si $\text{pgcd}(w,n) > 1$, nous avons un facteur premier de n.

Sinon, nous écrivons (n'oublions pas que d est connu) :

$$ed - 1 = 2^s r,$$

où $s \geq 1$ et r est impair. Nous savons que :

$$w^{2^s r} = w^{ed-1} = w^{k\phi(n)},$$

et par suite :

$$w^{2^s r} \equiv 1 \quad (n).$$

Donc, il existe un plus petit t tel que $t \leq s$ et tel que :

$$w^{2^t r} \equiv 1 \quad (n).$$

Notons t_0 cet entier. Si $t_0 > 0$ définissons :

$$v_0 = w^{2^{t_0 - 1} r},$$

de telle sorte que :

$$v_0 \not\equiv 1 \quad (n)$$

et que :

$$v_0^2 \equiv 1 \quad (n).$$

Maintenant si :

$$v_0 \not\equiv -1 \quad (n),$$

nous avons trouvé une racine carrée non triviale de 1 modulo n.

L'algorithme échoue, et nous devons le réappliquer à un autre w dans les deux cas suivants :

a) $t_0 = 0$, c'est-à-dire :

$$w^r \equiv 1 \quad (n);$$

b) il existe t tel que $0 \leq t \leq s-1$ et tel que :

$$w^{2^t r} \equiv -1 \quad (n).$$

Le nombre des w pour lesquels cet algorithme échoue est d'après le théorème de Rabin (voir [1], annexe B, p. 364) majoré par $\frac{\phi(n)}{4}$. Donc la probabilité d'obtenir le résultat avec un w donné est $\geq 3/4$. Si nous ne pouvons pas obtenir le résultat avec ce w on en tire un autre. La probabilité d'un nombre infini d'itérations est nulle. De plus l'espérance du nombre d'itérations est $\leq 4/3$. Maintenant nous pouvons conclure sachant que chaque itération a un coût polynomial.

Remarque importante : L'algorithme permet aussi de déterminer si la valeur de d est la bonne ou non.

13.4 L'attaque

On sait que :

$$ed \equiv 1 \quad (n),$$

c'est-à-dire :

$$ed = 1 + k\phi(n),$$

ou encore :

$$ed - k\phi(n) = 1.$$

Compte tenu des résultats connus sur la résolution d'une équation de Bézout (voir l'annexe B de [1], p. 346), on peut affirmer que puisque $0 < e < \phi(n)$ on a aussi $0 < k < d$.

En outre :

$$\phi(n) = (p-1)(q-1) = n - (p+q) + 1.$$

Donc :

$$ed = 1 + k(n - (p+q) + 1),$$

ce qui donne en divisant par dn les relations successives suivantes :

$$\frac{e}{n} = \frac{k}{d} + \frac{1 + k - k(p+q)}{dn},$$

$$\left|\frac{e}{n} - \frac{k}{d}\right| = \frac{k(p+q) - k - 1}{dn},$$

$$\left|\frac{e}{n} - \frac{k}{d}\right| < \frac{k(p+q)}{dn},$$

$$\left|\frac{e}{n}-\frac{k}{d}\right| < \frac{kq(\frac{p}{q}+1)}{dn},$$

$$\left|\frac{e}{n}-\frac{k}{d}\right| < \frac{3kq}{dn},$$

$$\left|\frac{e}{n}-\frac{k}{d}\right| < \frac{3k}{d\sqrt{n}},$$

$$\left|\frac{e}{n}-\frac{k}{d}\right| < \frac{3}{\sqrt{n}}.$$

Donc si :

$$d \leq \frac{n^{0.25}}{\sqrt{6}},$$

alors :

$$\frac{1}{2d^2} \geq \frac{6}{2\sqrt{n}},$$

et en conséquence :

$$\left|\frac{e}{n}-\frac{k}{d}\right| < \frac{1}{2d^2}.$$

D'après un résultat sur les fractions continues (voir le théorème C.14 de [1], p. 387), on en conclut que $\frac{k}{d}$ est une réduite de $\frac{e}{n}$.

L'attaque est alors la suivante : e et n sont publics, donc l'attaquant peut développer $\frac{e}{n}$ en fraction continue. Il teste alors pour toutes les réduites successives, si le dénominateur d permet ou non de factoriser n, et s' arrête dès qu'il a trouvé le bon d et donc la factorisation de n.

13.5 Les procédures utiles

```
##################
## alea(x)
##################

#### entree : nombre de bits x
#### sortie : nombre aleatoire s de x bits exactement

alea:=proc(x)
  local n,s;
```

```
  s:=1;
  for n from 1 to x-1
    do
      s:=rand(2)+2*s;
    od;
  RETURN(s);
end;

##################
## toutalea(x)
##################

#### entree : nombre de bits x
#### sortie : nombre aleatoire s de x bits au plus

toutalea:=proc(x)
  local n,s;
  s:=0;
  for n from 1 to x
    do
      s:=rand(2)+2*s;
    od;
  RETURN(s);
end;

##################
## nbpgene(x)
##################

#### entree : nombre de bits x
#### sortie : nombre premier p de x bits exactement

nbpgene:=proc(x)
  local p,a;
  p:=0;
  while (p=0)
    do
      a:=alea(x);
```

```
      p:=nextprime(a);
      if p>=2^x
        then p:=0;
      fi;
    od;
  RETURN(p);
end;


##################
## bezoutpq(u)
##################

#### entree : u tableau contenant la description d'un système RSA
#### sortie : v tableau contenant a et b tels que ap+bq=1

bezoutpq:=proc(u)
  local v;
  v:=iabcuv(u[1],u[2],1);
  RETURN(v);
end;

##################
## rsadcourt(x)
##################

#### entree : nombre de bits x
#### sortie : taleau u=[p,q,n,eul,ee,d]
####          où p et q ont x bits n=pq,
####          eul=(p-1)*(q-1), ee exposant de chiffrement
####          d exposant de déchiffrement (ou de signature).
####          Ici la taille de d est 1/8 de celle de n.


rsadcourt:=proc(x)
  local u,p,q,n,eul,ee,d,d1:t;
  p:=nbpgene(x);
  q:=nbpgene(x);
```

```
  n:=p*q;
  eul:=(p-1)*(q-1);
  while (ee=0)
    do
      ee:=toutalea(iquo(x,4));
      if ((ee>=eul) or (gcd(ee,eul)<>1))
        then ee:=0;
      fi;
    od;
  t:=iabcuv(eul,ee,1);
  d:=t[2];
  if d<0
    then d:=d+eul;
  fi;
  d1:=d;
  d:=ee;
  ee:=d1;
  u:=[p,q,n,eul,ee,d];
  RETURN(u);
end;

##################
## racun(n,ee,d,k)
##################

#### entree : module n, exposants ee et d
####          de chiffrement et de déchiffrement,
####          k, la taille de nombres w tirés au sort
####          cette taille est inférieure ou égale à celle de n.
####          Remarque : la procedure suppose d impair.
#### sortie : un tableau contenant les deux facteurs p et q de n
####          (la connaissance de d permet de factoriser n).
####          Remarque : si le d n'est pas le bon la procedure
####          renvoie [1,1].

racun:=proc(n,ee,d,k)
  local i,s,r,w,good,t,v,v0,u,pq;
  i:=0;
```

```
  good:=1;
  while (good=1)
    do
      w:=n;
      while (w >=n)
        do
          w:=toutalea(k);
        od
      r:=ee*d-1;
      s:=0;
      while (irem(r,2)=0)
        do
          r:=iquo(r,2);
          s:=s+1;
        od
      t:=0;
      u:=r;
      v0:=0;
      v:=powmod(w,u,n);
      while ((v<>1) and (t<=s))
        do
          v0:=v;
          v:=powmod(v,2,n);
          t:=t+1;
        od
      if ((t>0) and (v0<>n-1))
        then good:=0;
      fi;
    od
  pq:=[gcd(v0+1,n),gcd(v0-1,n)];
  RETURN(pq)
end;

#######################
## contfraction(n,ee,k)
#######################

#### entree : module n, exposant de chiffrement ee
```

```
####          c'est-à-dire la clé publique, la longueur choisie k du
####          développement en fraction continue.
#### sortie : tableau c des réduites successives du développement
####          de ee/n en fraction continue.

contfraction:=proc(n,ee,k)
  local v,c,p1,p2,q1,q2,p2s,q2s,pq;
  v:=dfc(ee/n,k);
  p1:=v[1];
  p2:=v[1]*v[2]+1;
  q1:=1;
  q2:=v[2];
  c:=[p1/q1,p2/q2];
  for i from 3 to length(v)
    do
      p2s:=p2;
      p2:=v[i]*p2+p1;
      p1:=p2s;
      q2s:=q2;
      q2:=v[i]*q2+q1;
      q1:=q2s;
      c:=concat(c,p2/q2);
    od;
  RETURN(c);
end;

#######################
## retrouved(n,ee,c,k)
#######################

#### entree : module n, exposant de chiffrement ee
####          c'est-à-dire la clé publique, c tableau de réduites,
####          k taille des w pour les tests de factorisation (procedure racun).
#### sortie : tableau w contenant deux objets :
####          le tableau pq contenant les deux facteurs
####          p et q de n, et i le nombre d'itérations nécessaires.

retrouved:=proc(n,ee,c,k)
```

```
  local i,s,OK,v,q1;
  s:=length(c);
  OK:=1;
  i:=2;
  while ((OK=1) and (i<=s))
    do
      q1:=denom(c[i]);
      if (irem(q1,2)=1) then
        v:=racun(n,ee,q1,k);
        if v[1]>1 then
          OK:=0;
        fi;
      fi;
      i:=i+1;
    od;
  w:=[v,i];
  RETURN(w);
end;
```

13.6 Une session de calcul

on lance les opérations suivantes :

```
u:=rsadcourt(512);
n:=u[3];
ee:=u[5];
d:=u[6];
c:=contfraction(n,ee,100);
w:=retrouved(n,ee,c,1023);
result:=w[1][1]*w[1][2]-n;
i:=w[2];
```

et on obtient :

$$u := rsadcourt(512) = [p, q, n, eul, ee, d] =$$

[115575051266682630132480239967727278608

618052844527795695878314766350487809699435652871645258993843151310631401856422634159784642859139332541986077162413 27,

1298798676767627210194987643521609081364
3650212656148288042839836346078501707535
5101925639817521670476490060293355850432
27423883557347382786263908859127407,

1501087236525180771002493004866838159709
4004851340369768340978551089436486027532
0914369394851882605495302846809417226345
1743792866949436743579956877554291327026
1406940237080569250713514671725872905221
6625811035983987199056681889790739096064
1333198734726963899437574877054191720015
42296637496548390519051749089,

1501087236525180771002493004866838159709
4004851340369768340978551089436486027532
0914369394851882605495302846809417226345
1743792866949436743579956877554291081571
2217505783569049460670315789858422359671
9516883178353315886075554607113260020584
4048122219213336098745879664781125285991
94453376180507928002476380356,

98135001339303405105892835074164108452943749895901597380646757917038437088297628675069124748156221879804352202060420679300273095773018179899200743906012527502955777978538621744751019851088747957270570639720457334778274834196081798174842565441683640278209491034172304692145353931081901426320355849660556810783,

179514730448291132293582318630185914587]

$$n := u[3] =$$

150108723652518077100249300486683815970940048513403697683409785510894364860275320914369394851882605495302846809417226345174379286694943674357995687755429132702614069402370805692507135146717258729052216625811035983987199056681889790739096064133319873472696389943757487705419172001542296637496548390519051749089

$$ee := u[5] =$$

9813500133930340510589283507416410845294
3749895901597380646757917038437088297628

$$6750691247481562218798043522020604206793$$
$$0027309577301817989920074390601252750295$$
$$5777978538621744751019851088747957270570$$
$$6397204573347782748341960817981748425654$$
$$4168364027820949103417230469214535393108$$
$$190142632035584966055681078 3$$

$$d := u[6] =$$

$$17951473044829113229358231863018591458 7$$

$$c := contfraction(n, ee, 100) =$$

$$[0, 1, \frac{1}{2}, \frac{2}{3}, \frac{15}{23}, \frac{17}{26}, \frac{287}{439}, \frac{1452}{2221}, \frac{3191}{4881}, \frac{20598}{31507}, \frac{23789}{36388},$$
$$\frac{44387}{67895}, \frac{112563}{172178}, \frac{944891}{1445319}, \frac{2947236}{4508135}, \frac{9786599}{14969724}, \frac{12733835}{19477859},$$
$$\frac{264463299}{404526904}, \frac{277197134}{424004763}, \frac{8303180185}{12700665031}, \frac{16883557504}{25825334825}, \frac{126488082713}{193478008806}, \frac{269859722930}{412781352437},$$
$$\cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots,$$
$$\frac{117359457094232776539145775550495740145}{179514730448291132293582318630185914587},$$
$$\cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots,$$

$$w := retrouved(n, ee, c, 1023) =$$

$$[[1298798676767627210194987643521609081364$$
$$3650212656148288042839836346078501707535$$
$$5101925639817521670476490060293355850432$$
$$27423883557347382786263908859127407,$$

$$
\begin{aligned}
&1155750512666826301324802399677272786 08\\
&618052844527795695878314766350487809699 4\\
&356528716452589938431513106314018564226 3\\
&415978464285913933254198607716241327], 79]
\end{aligned}
$$

On vérifie qu'on a bien factorisé n :

$$result := (w[1])[1](w[1])[2] - n = 0$$

On affiche le nombre d'itérations :

$$i := w[2] = 79.$$

Références

[1] **Pierre Barthélemy, Robert Rolland, Pascal Véron.** *Cryptographie : Principes et Mises en œuvre.* Hermes 2005

[2] **Bernard Parisse, Renée De Graeve.** *Paquetage logiciel Xcas.* http ://www-fourier.ujf-grenoble.fr/ ∼ parisse/giac_fr.html